Frédéric Boulanger, Daniel Krob, Gérard Morel and Jean-Claude Roussel, editors

Proceedings

Poster Workshop of the Complex Systems Design & Management Conference CSD&M 2014

Paris, France, November 12th 2014

A workshop at

CSD&M 2014

Organized by

CESAMES
Center of Excellence on Systems Architecture, Management, Economy and Strategy

Editor's addresses:

Frédéric Boulanger
Supélec
frederic.boulanger@supelec.fr

Daniel Krob
C.E.S.A.M.E.S. & École Polytechnique
dk@lix.polytechnique.fr

Gérard Morel
Université de Lorraine
gerard.morel@univ-lorraine.fr

Jean-Claude Roussel
Airbus Group Innovations
Jean-Claude.Roussel@eads.net

Contents

Preface

This volume contains the proceedings of the poster workshop at CSD&M 2014.

This workshop was organized to foster discussions about topics presented in papers that were not advanced enough to be published at the main CSD&M conference, but were worth a shorter presentation around a poster.

The program committee of CSD&M selected these papers for presentation at the Poster Workshop and publication in separate proceedings.

About CSD&M

The purpose of the "Complex Systems Design & Management" (CSD&M) conference is to be a forum for both academic researchers and industrial actors working on complex industrial systems architecture and engineering in order to facilitate their *meeting*.

The CSD&M academic–industrial integrated dimension

To make the CSD&M conference this convergence point of the academic and industrial communities in complex industrial systems, we based our organization on a principle of *complete parity* between academics and industrialists. This principle was first implemented as follows:

- the Program Committee is 50% academics and 50% industrialists,
- Invited Speakers are coming in a balanced way from numerous professional environments.

The set of activities of the conference followed the same principle. They indeed consist of a mixture of research seminars and experience sharing, academic articles and industrial presentations, software and training offers presentations, etc. The conference topics cover in the same way the most recent trends in the emerging field of complex systems sciences and practices from an industrial and academic perspective, including the main industrial domains (transport, defense & security, electronics & robotics, energy & environment, health & welfare services, media & communications, e-services), scientific and technical topics (systems fundamentals, systems architecture & engineering, systems metrics & quality, systemic tools) and system types (transportation systems, embedded systems, software & information systems, systems of systems, artificial ecosystems).

August 25, 2014
Gif-sur-Yvette

Frédéric Boulanger
Daniel Krob
Gérard Morel
Jean-Claude Roussel

Taming the Complexity of Big Data Multi-Cloud Applications with Models

Marcos Aurélio Almeida da Silva[1], Andrey Sadovykh[1], Alessandra Bagnato[1], Etienne Brosse[1]

[1] R&D Department, SOFTEAM, 9 Parc Ariane, Guyancourt, France
marcos.almeida@softeam.fr, andrey.sadovykh@softeam.fr, alessandra.bagnato@softeam.fr, etienne.brosse@softeam.fr

Abstract. Private and public clouds are getting more and more common. With them comes the need to analyze data stored by different applications in different clouds. Different clouds and applications tend to enforce the use of different data stores, which makes it even harder to aggregate information. The main outcome is that integrating different data sources requires deep knowledge on how data is stored on each solution and on the trade-offs involved in moving from one system to another. This paper is part of the ongoing work on the JUNIPER FP7 EU project (http://www.juniper-project.org/). In that project we explore the power of modelling tools to simplify the design of industrial big data applications. In the present work we present an overview of our approach and its application on a simple case study.

1 Introduction

The advent of cloud computing and the multiplication of computing and storage power available to companies gave rise to the so-called **big data cloud applications**. The multiplication of cloud offers also lead to a fragmentation in the capabilities of different providers [1]. Similarly, the multiplication of data management tools lead to the fragmentation of the data representation paradigms.

Concerning the fragmentation of cloud providers. On the one hand, some companies are surely in a position to take advantage of that. On the long run, applications end up with a set of specialized applications, each of them based on a different stack of tools. The challenge to these companies is then in aggregating the data stored on different stores, on different providers and behind different systems to support their business decisions. This challenge stems from the fact that highly specialized developers are needed to deal with the different stacks and programming languages. Connecting them therefore becomes more expensive and complex the more languages and systems a company needs to integrate.

In this paper we investigate the strengths of **model driven engineering (MDE)** in such scenario. Model driven engineering in fact consists in using high level models to abstract the complexity in an application. MDE techniques excel particularly in dealing with multiple programming languages and frameworks [2] [3]. This is so because model transformations can encapsulate the complexity involved in the translation of

high level concepts into concepts in other languages, and therefore reduce the necessity of highly specialized developers and then the cost of integrating disperse data stores. Models are also useful when dealing with availability time and consistency constraints by means of model analysis. This is so because high level models on a single language are much easier to inspect and analyse (either automatically or manually) than multi-language code.

The main challenge in applying MDE to big data applications relies in **finding the right** abstract **modelling language** that is **rich enough** to abstract from the specific features of the connected platforms, but still **low level enough** to foster code generation. In the context of big data application this is even harder because, to the best of our knowledge, no language exists that includes both the concepts necessary to define the architecture of a cloud application along with its data streaming and analysis in a high level. This is therefore the main contribution of this paper.

In this paper we present the approach developed as part of the JUNIPER FP7 EU project. It consists in identifying a subset of the Unified Modelling Language (UML) and extending it with data analysis and processing concepts so that it is suitable to both defining the architecture of a big data cloud application and to generate code for it. Since the project is in the beginning of its second year, in this paper we present the initial insights and experiments behind the use of a MDE approach to support the design of multi-cloud big data applications. In order to avoid exposing sensitive details of the project use cases, in this paper we apply our approach to a similar application based on the same principles.

This paper is structured as follows: Section 2 presents an overview of multi-cloud big data applications and the requirements of a MDE approach for it, Section 3 presents the approach we put forward in this paper. Section 4 presents a case study involving a multi clouds big data application. Section 5 presents the related work and Section 6 finally concludes.This is your introduction.

2 Multi-cloud Big data applications & Model Driven Engineering

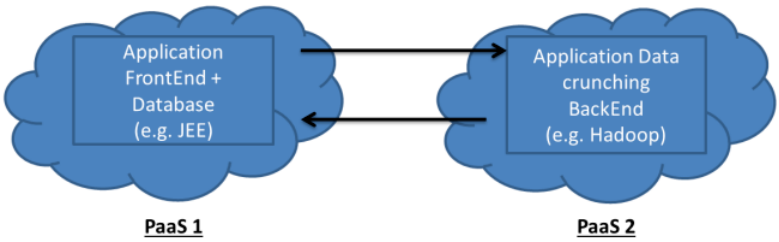


Fig. 1. Typical example of multi-cloud big data application.

Fig 1. illustrates a typical multi cloud big data application. It consists in leveraging the stronger points of different cloud offers to build a complex application. In this example, we consider a company that wants to leverage the cloud computing Platform As A Service (PaaS). The cloud application this company wants to build consists of a classic front end along with a database application and a specific back end for data

crunching. This backend is used to process the data in the application to deliver new insights.

The requirements in terms of short and long term data storage caps, response time and programming framework for each part of the application are different. That is therefore an **opportunity** for this company in using different PaaSes, in order to **optimize the costs** of providing the application.

The main challenge in seizing this opportunity is however in dealing with the **heterogeneity** of the cloud providers. In this specific example, the company would need to deal with the different programming languages and frameworks supported by each PaaS (e.g. Java JEE in the front end and Java Hadoop for the data crunching backend). While the application may be cheap to produce and deploy in the short term, the multiplicity of backend technologies may be a threat to its future maintenance [4].

In this paper we leverage a **model driven engineering** based approach to **seize this opportunity**. A MDE-based approach allows companies to increase the level of abstraction of architecture of the application by means of a model. This model can then be used to **simplify the development** by reducing the required familiarity of developers with the PaaSes frameworks and **maintenance** tasks by simplifying tasks such as moving data and code to other PaaSes.

There are however two challenges into coming up with a MDE approach for multi cloud big data applications:

1. **What language should we use to model applications?**
   The challenge here consists in using a language that includes the abstraction related to cloud applications, big data and the deployment of such application in multi-clouds.
2. **How to make sure this language is low level enough to foster code generation?**
   The challenge here lies in supporting as much as platforms as possible so that one can make sure that the models correspond to the effectively deployed applications.

3 A UML BASED MDE approach for big data cloud applications

3.1 General Approach

As explained in Section 2, the two challenges underlying the use of MDE approaches for handling the problem of designing multi-cloud applications are: (i) choosing a modelling language that is high-level enough to abstract from the concepts in different programming languages, and (ii) low level enough to allow for code generation. We chose the Unified Modelling Language [5] as modelling language since its object-oriented roots have been shown to be useful to model a wide range of problems while still serving as basis for code generation.

It's main drawback is however in the complexity of its specification, and therefore the steep learning curve that it represents to developers. Our approach to countering

this drawback consists in selecting a **subset** of UML suitable to address both challenges. The subset selected in this paper starts with the subset of the language usually reused by code generation tools [6] [7] [8] [9] [10].

However, reusing UML is not enough: one still needs to represent multi-cloud and big data specific concepts. In order to do that, we use the standard extension mechanism of UML called UML profiles. A UML profile allows one to add new concepts to the language by extending existing ones. On top of the extended language, we implemented code generators, so that the maps such abstract concepts into working code.

On the next section, we present the subset of UML we reuse, and the new concepts we add to it.

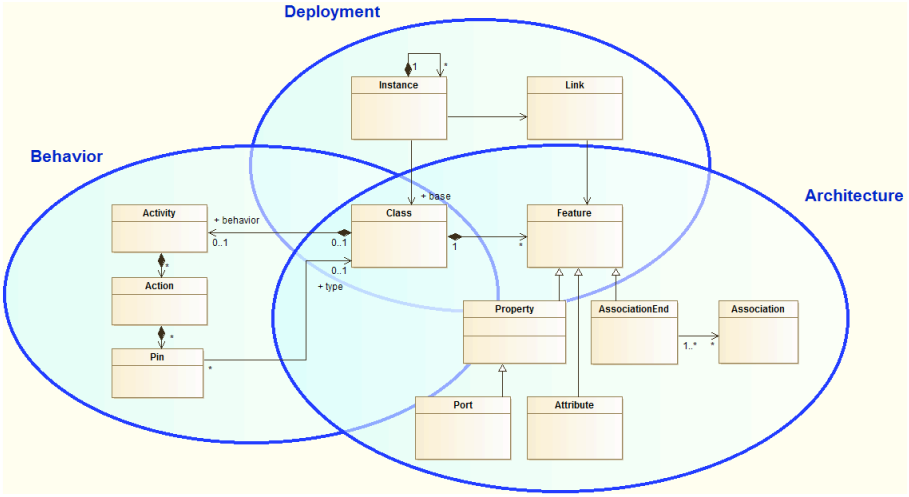


Fig. 2. Simplified overview of the reused UML subset.

3.2 UML for modelling multi-cloud applications & data

Fig 2. shows the subset of UML we reuse to model big data applications. In order to represent the **architecture of the application**, we reuse the concepts in UML class diagrams, i.e. applications are represented as `Classes` along with their `Properties` and `Associations`. `Classes` are also used to represent data types in a cloud provider independent way. Still when it comes to representing big data, one needs to represent the **data flows** that need to be implemented by the application.

We reuse the UML activity diagram concepts in order to represent the data flow, i.e. `Activities` which are broken down into atomic `Actions`. `Actions` have input and output `Pins`, which describe its input and output parameters.

The deployment of the application is represented by a set of UML object diagram concepts, featuring `Instances`, their relationships (`Links`) and their `base` types.

Stereotype	Input(s)	Output
filter, split, limit	Filtering expression.	Split data streams, or filtered data stream, or limited subset of the data stream.
generate	A set of streams and generation expression.	A data stream obtained by application of the generation expression to the input streams..
group	A data stream and a grouping criteria.	A data stream formed by groups obtained by application of the grouping criteria.
union	A set of data streams.	A single data stream containing both inputs.
cross	A set of data streams.	The cross product of the input streams.
inner-join, outer-join	A set of data streams and joining expressions.	A joined data stream.
sample	The size and type of sample to generate.	A randomly generated data stream.
order, distinct	A data stream.	An ordered data stream or a data stream containing only distinct elements.
load, store	A data stream.	Loads or saves the data stream to a persistency medium.

Fig. 3. Big Data flow language. Input and output parameters should be rendered as UML input and output Pins.

3.3 UML profile for representing complex big data flows

An object diagram is made **cloud specific** by conforming to a **fixed structure**. At the top level, instances represent **multiple cloud providers**. These instances contain other instances that represent either resources provided by the cloud provider in order to deploy parts of the application (most commonly on IaaSes) or parts of the application itself (most commonly on PaaSes). This way, object diagrams can represent the required resources from different clouds, and their interconnection.

It is important to notice that UML concepts are not sufficient to represent to the full extent the details of the architecture, deployment and behaviour of a cloud application. For the sake of brevity, in this paper we only present the concepts necessary to extend activity diagrams for representing complex big data flows. Extensions of UML for representing complex cloud application architecture and deployment are part of our ongoing work on the FP7 EU projects REMICS[1] and MODAClouds[2].

The UML activity diagram concepts are however not sufficient to represent all big data flow concepts. That is why we extended them. Some of the added concepts are represented in **Fig. 3**. These stereotypes were based on the concepts behind the PigLatin language [11], which abstracts the data processing works on top of the Ha-

[1] http://www.remics.eu/
[2] http://www.modaclouds.eu/

doop framework. They extend the UML basic concept of `Action` providing a big data flow processing specific semantic to them.

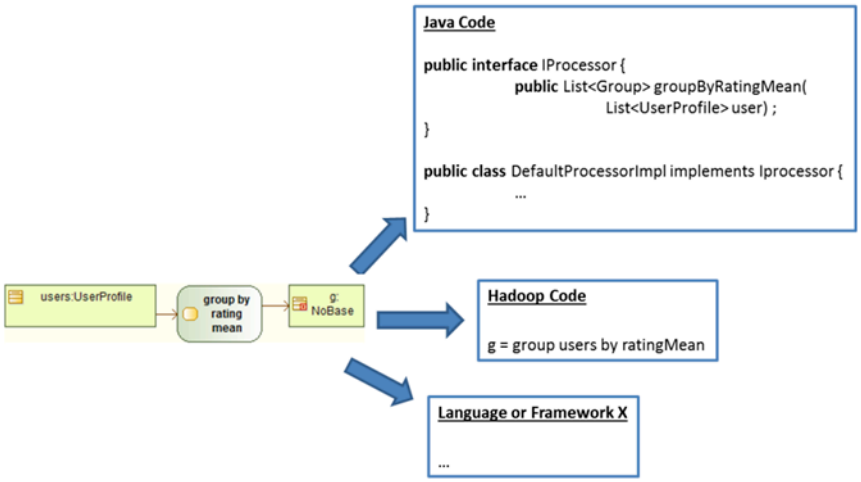


Fig. 4. Code generation approach.

As we discussed in Section 2, when it comes to supporting multi-cloud big data applications, being able to support code generation is as important as supporting modelling of such systems. **Fig. 4** overviews our code generation approach aiming to achieve this objective.

The code generation approach consists in providing cloud generators upfront, along with the definition of the language. In our present case, we experimented with code generation for Java EE cloud applications and Hadoop PigLatin map reduce data flows. As illustrated in **Fig. 4**, we use the same UML model, along with the provided transformations to generate both target languages.

4 Case Study

In this section we put the models and transformations we defined in the previous section into action. Notice however that this is a proof of concept that illustrates the work that is being performed on the FP7 EU project JUNIPER case studies. To avoid publishing sensitive information, we base the present case study in a cloud application that can be found in the literature. It consists in the MiC application (Meeting in the Cloud) [12]. The modelling and code generation tools presented here were implemented using the Modelio modelling tool[3].

The MiC application is a social network which allows users to maintain user profiles in which they register they topics of interests. The MiC application then groups

[3] Modelio, the open source modelling environment. Website: http://www.modelio.org

users by similarity, allowing users to interact with their "best contacts", based on ratings provided by each user in their profiles.

The use case we will analyse here is the one of a company that intends to provide different levels of service for different categories of users. Some users have paying accounts while other have free accounts. The company providing the MiC service wants then to support updates to the similarity computing service to paying users as fast as possible. In order to do that, it will use a IaaS cloud provider to compute the similarity so that the resources of the provider may adapt to the number of user requests and therefore absorb any increase in demand. For free account users, the company will resort to a low cost PaaS. This will lead to minimum cost and management costs.

The challenge is that the selected clouds support different platforms: on the IaaS the company may use any technology, while on the low cost PaaS, it needs to use the Hadoop crunching platform as a limitation of the PaaS.

This section intends to show how a MDE approach can help the company in designing the application so that it can be deployed on both clouds. This section is divided into two sections, in the first one, we show the UML models that describe the MiC applications and the second the code generation techniques we developed from these models.

4.1 UML Models

Fig. 7 and **Fig. 6** display part of the UML models that describe the MiC application. **Fig. 7** is divided into three parts:

- The **data model** is centred on the `UserProfile` class that represents a profile on the system, and stores the `UserRatings` provided by the user and the `UserSimilarity` which group profiles by similar ratings.
- The **architecture model** shows that there are basically two components in the application: the `CRUD` which is responsible for displaying the CRUD user interface and the `SimilarityCimputer` to update similarity of the users.
- Finally, the **deployment model** states that part of the application is deployed on `iaas1` and part of it on `paas1`. The PaaS is used to compute the similarity of part of the user base.

Fig. 6 models part of the behaviour of the `SimilarityComputer` component. It consists in loading user profiles and ratings from the data base, joining them, computing the means of the ratings and then grouping profiles by similarity.

4.2 Code Generation

Fig. 8 displays the generated code for the model described in the previous subsection. On the right side we see the code generated for the Hadoop cloud and on the left side we see the code generated for the Java EE Cloud. Thanks to the MDE approach, developers can be sure to find the same behaviour on both clouds. Data types and

structures are translated accordingly and the code generation makes sure that the data flow of both implementations is similar.

In case a new cloud platform is added on the future, developers will need to spend less time re-implementing the same data flow and data structures on the new cloud.

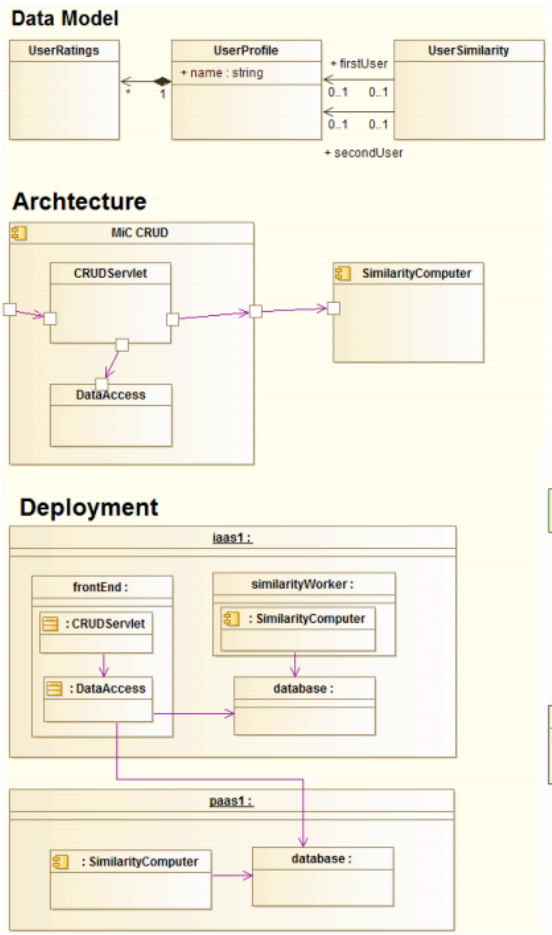


Fig. 7. Overview of the MiC Application models

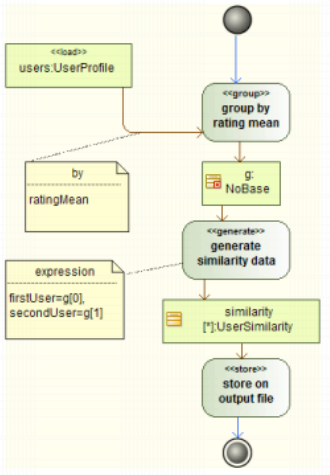


Fig. 6. Similarity Computer Behaviour model

Code for Java EE Cloud

```
@Entity
public class UserProfile {
…
}
@Entity
public class UserRatings {
…
}
public class ProcessSimilarity {
…
   List<Group> groupByRatingMean(
List<UserProfile> users) {
   …
```

Code for Hadoop Cloud

```
users = load 'input.csv' using PigStor-
age(',') s
   (logn:chararray,
      meanRating:float);

g = group users by ratingMean;

similarity = foreach g
   generate (first);

store similarity
   into 'output.csv'
              using PigStorage(',');
```

Fig. 8. Generated Code.

5 Related Work

In this section we are going to study similar works in the literature that are used to model applications, either related to the high level architecture of the application, its deployment or data. We are also going to analyse their support concerning the generation of running code from the models, and therefore of being suitable to a MDE approach for big data multi-cloud applications.

5.1 Architecture modelling languages

Languages such as SoaML [13]and SoaMF [14] are used to define the high level architecture of cloud applications. SoaML defines a MOF metamodel and a UML profile while SOMF defines a completely new language for defining service related concepts. It reuses and extends the UML concepts of components and ports to define respectively the services and their interfaces. SOMF also includes a sublanguage called Cloud Computing Modelling Notation (CCMN), whose concepts include IaaS, PaaS and SaaS clouds, and clouds of clouds; and service orchestration based.

The main weak point of these languages is that they are not intended to support code generation neither the data types manipulated by the application, they are limited therefore on high level concepts.

5.2 Cloud deployment languages

Some languages focus on providing "DevOps" tools such as Chef [15], Puppet [16] and CloudML [17]. They intend to automate the deployment of applications and services, as well as the management of cloud capabilities. With visibility and control on both IaaS and PaaS levels, developers can exploit the peculiarities of cloud solutions at each level of the cloud stack. The main weak point of such languages is that they target DevOps, not developers, and therefore do not include the architecture and data types of the application, and their impact to its execution.

Another group of languages uses the existing UML deployment diagrams to model the physical distribution of data. As an example we have the work of S. Lujan-Mora and J. Trujillo [18]. They define a UML profile that they can use to specify the deployment of Data Warehouses, which can be considered a former kind of private clouds, and could be extended to potentially represent multi-cloud big data deployment. Similarly, data integration tools such as Pentaho [19] and Yahoo! Pipes [20] offer visual editors that allow one to describe the partitioning of data in different data stores.

In both categories of work, the data structures are considered but not the architecture of the application, neither the code generation is envisioned.

5.3 Data Modelling

Many tools and approaches exist to help data modellers and application developers to describe data models. Object oriented models can be produced with the help of languages like UML [5] or Entity Relationship models [21], and relational models can be produced with the help of the numerous UML Profiles for relational modelling [22] [23] [24] [25] [26]. Purely object oriented databases are however rarely used in practice when it comes to storing data. Translations between both paradigms were created with the purpose of facilitating the use of relational data stores by object oriented applications [22] [27] [28]. This comes with the drawback of the inherent loss of information in the translation process. As in the other cases, these languages do not support modelling the application platform neither its deployment on the cloud.

6 Conclusion

On the one hand the multiplication of cloud providers represents an opportunity to companies willing to reduce the costs of maintenance of cloud applications by choosing the set of providers that best adapts to the uses of the application. On the other hand, multiple cloud providers come with extra technical requirements on programming languages, data structures and framework support. Integrating data and applications from different clouds then becomes more and more expensive and complex as the number of cloud providers increases.

In this paper we presented the first steps in dealing with this problem by means of a MDE approach. The core of the approach consists in defining a language, based on a

UML subset, extended with big data analysis specific concepts that can be used to generate multi-cloud enabled code to aggregate data in different sources. This approach is going to be fully implemented in the foregoing year of the JUNIPER FP7 EU project and will be applied on two industrial case studies.

Acknowledgements

The research reported in this article is partially supported by the European Commission grant no. FP7-ICT-2011-8- 318763 (JUNIPER).

References

1. R. Prodan and S. Ostermann, "A survey and taxonomy of infrastructure as a service and web hosting cloud providers," in *10th IEEE/ACM International Conference on Grid Computing*, 2009.
2. P. Baker, S. Loh and F. Weil, "Model-Driven Engineering in a Large Industrial Context: Motorola Case Study," in *Model Driven Engineering Languages and Systems*, Springer Berlin Heidelberg, 2005, pp. 476-491.
3. F. Fleurey, E. Breton, B. Baudry, A. Nicolas and J.-M. Jézéquel, "Model-Driven Engineering for Software Migration in a Large Industrial Context," in *Model Driven Engineering Languages and Systems*, Springer Berlin Heidelberg, 2007, pp. 482-497.
4. T. Mens, M. Wermelinger, S. Ducasse, S. Demeyer, R. Hirschfeld and M. Jazayeri, "Challenges in software evolution," in *Eighth International Workshop on Principles of Software Evolution*, 2005.
5. OMG, "UML: OMG Unified Modeling Language (OMG UML) Superstructure, Version 2.4.1," 2001.
6. "ArgoUML Code Generation Window," [Online]. Available: http://argouml.tigris.org/tours/classgen.html. [Accessed 14 January 2014].
7. [Online]. Available: http://www.nomagic.com/products/magicdraw.html.
8. "IBM," [Online]. Available: https://www-304.ibm.com/support/docview.wss?uid=swg21259513. [Accessed 14 January 2014].
9. "Modelio," [Online]. Available: http://www.modeliosoft.com/en/modelio-store/modules/generators/java-designer-open-source.html.
10. "Uml to Java Generator 2.0.2," [Online]. Available: http://marketplace.eclipse.org/content/uml-java-generator#.UtPwpfRDvfk. [Accessed 14 January 2014].
11. "Pig Latin Reference Manual," [Online]. Available: http://pig.apache.org/docs/r0.7.0/piglatin_ref1.html. [Accessed 14 January 2014].

12. G. F., L. D., S. Y. M., A. D. and D. N. E., "An Approach for the Development of Portable Applications on PaaS Clouds," in *Proceedings of the 3rd International Conference on Cloud Computing and Service Science (CLOSER 2013)*, 2013.
13. OMG, "Service oriented architecture Modeling Language (SoaML)," 2009..
14. B. Michael, "Introduction to Service-Oriented Modeling," in *Service-Oriented Modeling: Service Analysis*, Wiley & Sons.
15. "Chef," [Online]. Available: http://www.opscode.com/chef/.
16. [Online]. Available: https://puppetlabs.com/.
17. [Online]. Available: http://cloudml.org/.
18. J. T. Sergio Luján-Mora, "Physical Modeling of Data Warehouses Using UML Component and Deployment Diagrams: Design and Implementation Issues," *Database Management,* pp. 12-42, 2006.
19. [Online]. Available: http://www.pentaho.com/.
20. [Online]. Available: http://pipes.yahoo.com/pipes/.
21. C. Batini, S. Ceri and S. B. Navathe, Conceptual Database Design, an Entity-Relationship Approach, Benjamin and Cummings Publ. Co., 1992.
22. [Online]. Available: http://argouml-db.tigris.org/.
23. Rational, "The UML and Data Modeling," [Online]. Available: http://bit.ly/15hxqgj.
24. [Online]. Available: http://www.modeliosoft.com/en/modules/sqldesigner.html.
25. D. Gorni, "UML Data Modeling Profile," 2002. [Online]. Available: http://bit.ly/VYJVGw.
26. D. Silingas and S. Kaukenas, "Applying UML for Relational Data," 2004. [Online]. Available: http://bit.ly/VkrNtw.
27. "Relational Persistence for Java and .NET," [Online]. Available: http://hibernate.org.
28. Oracle, "Java™ Persistence 2.0, JSR 317".
29. S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *SIGACT News,* pp. 51-59, 2002.

Scalability in System Design and Management, the MONDO Approach in an Industrial Project

Alessandra Bagnato[1], Etienne Brosse[1], Marcos Aurélio Almeida da Silva[1], Andrey Sadovykh[1]

[1] R&D Department, SOFTEAM, 9 Parc Ariane, Guyancourt, France
alessandra.bagnato@softeam.fr, etienne.brosse@softeam.fr, marcos.almeida@softeam.fr, andrey.sadovykh@softeam.fr

Abstract. The current system designs and management technologies are being stressed to their limits in terms of collaborative development, efficient management and persistence of large and complex models. As such, a new line of research is imperative in order to achieve scalability across the system design space. Scalability in system design has different dimensions: domains, team localizations, number of engineers, size and management of the engineering artefacts, interoperability and complexity of languages used. This paper depicts how the MONDO FP7 EU project (http://www.mondo-project.org/) aims to comprehensively tackle the challenge of scalability in system design and management by developing the theoretical foundations and an open-source implementation of a platform for scalable modelling and model management. An industrial case study is also presented. The system designed in this case study is distributed among several and dependent units, domains, and languages.

1 Introduction

As Model Driven Engineering (MDE) is increasingly applied to larger and more complex systems, the current generation of modelling and model management technologies have being pushed to their limits in terms of capacity and efficiency. Therefore additional research is imperative in order to enable MDE to keep up with industrial practice and continue delivering its widely recognized productivity, quality, and maintainability benefits.

In the following section, we will present how the MONDO project plans to handle the increasingly important challenge of scalability in MDE. In section 3, we present how some issue tackled by the MONDO [2] approach have already been implemented and used on an industrial project within the Modelio Modeling tool environment [1].

2 MONDO Approach

Achieving scalability in modelling and MDE involves being able to construct large models and domain-specific languages in a systematic manner, enabling teams of modelers to collaboratively construct and refine large models, advancing the state-of-the-art in model querying and transformation tools so that they can cope with large models (of the scale of millions of model elements), and providing an infrastructure for efficient storage, indexing and retrieval of large models. To address these challenges, MONDO will develop or optimize algorithms at different levels of the system modelling. Obviously techniques and tools will be implemented at model and model engineering levels. MONDO approach also takes into account at higher levels, depicted in **Fig. 1**, i.e. the meta-model engineering and meta-model levels.

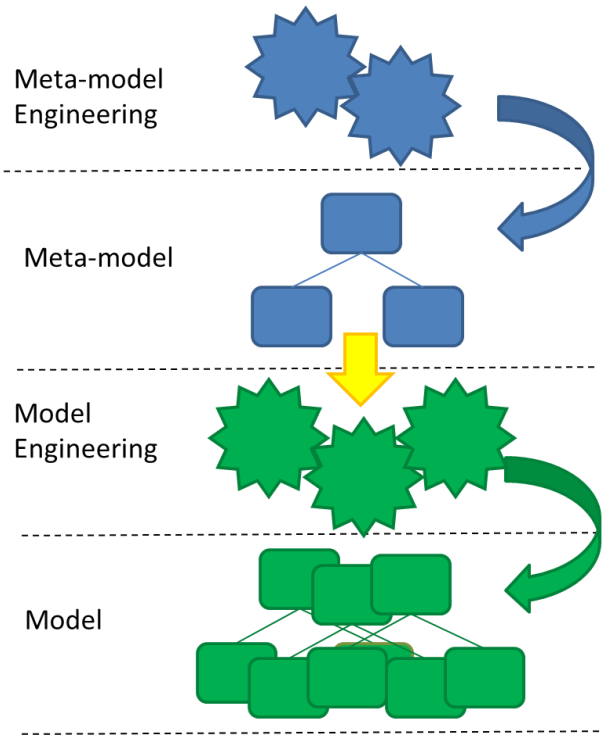


Fig. 1. The levels considered by the MONDO approach

In section 2.1, we describe the MONDO research field at the meta-model and meta-model engineering levels. Section 2.2 presents the MONDO targeted work at the model and model engineering levels. Finally sections 2.3 and 2.4 highlight that the techniques explored apply at multiple levels in the MONDO approach.

2.1 Scalability at the meta-model and meta-model engineering levels

The first objective of MONDO is to develop scalable techniques and processes for the engineering meta-model modelling or domain-specific languages (DSLs)[9]. Providing techniques for the efficient support and optimization of languages for which large models (instance of these meta-model or DSL) are expected. Also, the complexity or size concerns are in the definition of the languages themselves and hence we will propose methods; enable the engineering of such languages.
MONDO will complement these techniques with processes facilitating their use in complex projects dealing with large artefacts. In order to achieve these objectives, MONDO will provide techniques for developing languages enabling scalable modelling, techniques for developing languages enabling scalable modelling, scalable concrete syntaxes, scalable MDE processes. The following paragraphs and more in detail in [9] describe the technologies supported by the project.

Techniques for developing languages enabling scalable modelling.

MONDO will investigate techniques to construct an efficient infrastructure for DSLs that are expected to have large instance models. In this way, it will propose techniques to describe optimizations in the storage/retrieval of large instance models. These optimizations will be specified at the meta-model level, when the DSL is being built, and could be suggested by a recommender system. We will provide support for defining and reusing abstractions for DSMLs that will produce simpler views of large models. Another line of work will be the automated modularity support for DSLs, as well as the possibility to build large models by compositing reusable model fragments and template models from a library spanning across heterogenous technical spaces (DSLs, UML, Matlab/Simulink, etc).

Techniques for developing languages enabling scalable modelling.

Often, the definition of a DSL is itself complex, which renders its construction in an ad-hoc manner challenging. Therefore, we will provide methods and techniques to construct meta-models for large, complex DSLs. In particular, we aim to develop techniques for incremental, example-based construction of meta-models, supported by automated guidelines or meta-model patterns. We will also develop methods for meta-model construction by reusing existing fragments and meta-model templates from a library spanning across heterogenous technical spaces (DSLs, UML, XML schemas, etc).

Scalable concrete syntaxes.

MONDO will develop techniques for designing scalable visual concrete syntaxes. These techniques should enable the visualization and navigation of large models, including their visualization at different levels of detail or abstraction, in connection

with the defined abstractions at the abstract syntax level. Techniques will also be developed to automate the visualization and exploration of large scale models, for which no concrete graphical syntax has been defined. The techniques developed in this task will be delivered as ready-to-use library elements, enabling the rapid development of visual editors with built-in support for abstractions and facilities for model exploration.

Scalable MDE processes.

MONDO will investigate processes (and tool support for them) for the use of MDE in complex projects, dealing with large artefacts. Hence, we will bring ideas from agile development into MDE. For example, similar to continuous integration, we will verify the consistency of models on the server in the same way we run automated builds and unit tests. We will also apply "by-example" techniques to the incremental construction of large, complex DSLs, and adopt techniques and processes from reutilization-based development in order to characterize reusable assets and facilitate their reutilization.

2.2 Scalability of the model operations

Any non-trivial MDE project involves querying and manipulation of a substantial number of models. These model manipulation operations are usually implemented as model-to-model transformations that take as input one or more source models and generate as output one or more target models, where target and source models can conform to the same or to different meta-models. Model queries are primary means to extract views and to formally capture and validate well-formedness constraints, design rules and guidelines on the fly. Therefore, scalability of model queries and transformations is a key element in any scalable MDE solution. Our experience with industrial case studies is that current transformation technologies do not scale, which discourages some potential adopters from using MDE.

So the second objective of MONDO is to create a new generation of model querying and transformation technologies that can solve this problem. We propose to build a reactive transformation engine by combining incremental change propagation with lazy computation. We plan to provide the engine with strong parallelization properties, to be able to fully exploit distributed/cloud execution environments.

Re-evaluating a validation query or regenerating a full target model after a few local changes in the underlying (source) model can be particularly inefficient. An incremental query evaluation technique will propagate model changes directly to the affected queries to incrementally update their result set. An incremental transformation algorithm will minimize the number of transformation rules to be re-executed according to the changes on the source model, while ensuring the synchronicity between source and target models.

Lazy / on-the-fly / on-demand creation of target models.

Generating a full target model from a source model can be time consuming, especially since often only parts (in proportion to the full model size) of the target model will be accessed. A lazy evaluation algorithm would try to execute the transformation on-demand, only producing the subsets of the target model when they actually need to be accessed.

Queries and Transformations in the cloud: parallel/distributed execution Model queries and transformations can benefit from leveraging distributed cloud architecture for their execution. If adapted, queries and transformations can rely on the scalability of cloud architectures to deal with large models. Two kinds of adaptations can be performed: 1) partitioning of the model and parallel execution of the full transformation on each model slice and 2) partitioning of the transformation and execution of transformation subsets on different cloud nodes. In both cases the result must be merged at the end.

Infinite/streaming transformations.

In some MDE application scenarios we need to deal with infinite models (there can be a continuous influx of data into the model). Therefore, a transformation cannot wait until the model is complete to start transforming it; instead it needs to be able to output target model elements as source elements are becoming available. Traces between target and source elements have to be kept in memory until the system reaches a certain degree of confidence that they will not be needed for future computations.

Integration with scalable persistence mechanisms.

Here MONDO will develop interfaces that enable model management languages to query and modify models persisted using the technologies proposed in an efficient manner.

2.3 Collaborative modelling

The objective at this level is to provide new collaborative modeling tools for geographically distributed teams of engineers working on large-scale models using heterogeneous devices (desktop computers, laptops, tablets, mobile phones, etc). These collaborative modeling tools will allow multiple teams from different stakeholders (such as system integrators, subcontractors, certification bodies) to simultaneously access the server-side model as clients in a scalable and secure way respecting access control policies.

For this purpose, MONDO will first investigate and adapt offline (asynchronous, long transaction) and online (synchronous, short transaction) collaboration patterns for models. Offline collaboration (like SVN or CVS) is widely used in collaborative software engineering where developers commit a larger portion of changes as a long

transaction (e.g. at the end of the day). Online collaboration is popular in collaborative document authoring (analogously to GoogleDocs or LucidCharts) where a group of collaborators cooperatively and simultaneously edit the same document, and each individual change is immediately committed to a master copy.

For offline collaborative modeling, novel intelligent, user-guided techniques will be developed based upon design space exploration techniques to resolve conflicts upon concurrent commits. Furthermore, the work package will investigate how to determine the proper size and boundaries of model fragments to balance collaboration and performance of underlying queries and transformations. Intelligent, view-driven dynamic locking mechanisms will be proposed based upon incremental model queries to avoid unintentional concurrent changes of model elements, which is a key issue for both online and offline collaborative modeling. Secure model access control policies will be defined uniformly for online and offline collaboration, and will be enforced to facilitate collaboration between teams of different stakeholders.

A multi-device collaborative modeling framework will be developed to demonstrate the feasibility and scalability of different collaboration patterns over heterogeneous devices as collaboration means. An interface will be defined to encapsulate this collaborative modeling layer and will be offered as services to high-level domain-specific modeling, query and transformation tools. More details on the planned work on collaborative modeling can be found at [3] [4] [5] [6].

Primitives and Patterns for Collaborative Modeling.

Various collaboration primitives and patterns for offline (e.g. SVN-based) and online strategies (e.g. collaborative modeling authoring sessions) will be investigated. This task also aims to provide consistency management techniques for collaborative modeling adapted from version control systems including locking, transaction handling with commit, conflict management and resolution. Furthermore, this task also incorporates the development of a new version of a model by a team of developers during collaborative modeling sessions where the consistency of persistent storage can be temporarily violated to improve collaboration.

Secure Access Control for Collaborative Modeling.

Confidentiality support for collaborative modeling by providing secure access control will be also provided. Access control policies will be defined at a high level uniformly for online and offline cases, which will be mapped to the security means provided by the underlying model storage frameworks.

Interface for Collaborative Modeling.

At this point MONDO will define an interface for collaborative modeling services (including update, commit, lock, etc.), which will by used by advanced modeling and

transformation tools. This interface will be aligned with underlying scalable model storage mechanisms, and will offer means for both online and offline collaboration.

Multi-Device Collaborative Modeling Tool.

A multi-device collaborative modeling tool will be developed which allows various development teams to simultaneously and collaboratively access, query and manipulate the underlying models using heterogeneous devices as a collaboration means. Integration with existing domain-specific modeling tools and the persistence layer will be provided

2.4 Efficient model persistence

The aim of this level is to develop an efficient model persistence format in order to address the limitations of the current standards (e.g. XMI), and to deliver a scalable model indexing framework. This will enable the querying and transformation languages and the collaborative modelling tools developed in to efficiently manage large and heterogeneous models.

3 Industrial Case study

Not all points developed in the MONDO approach (cf. Section 2 MONDO Approach) have been implemented or even specified yet within the project being the project in its early stage. We detail in the following section the lazy loading of models solution that already exist within our Industrial Case Study and it is currently implemented within the Modelio Modeling environment [1] and that will serve as a case study to evaluate the MONDO project results.
The case study consists of a distributed modelling of a system between a heterogeneous set of users. For this we used on one hand the distributed framework - described in section 3.1 Modelio Distributed framework - and on the other one the view/viewpoint and lazy loading mechanism both of them provided by Modelio tool [1]. All the technologies provided by MONDO will be evaluated within the same industrial case study.

3.1 Modelio Distributed framework

The concept of the “modeling project” has been completely overhauled under Modelio[1]. A Modelio project now groups a certain number of local or remote models. Local models, or working models, can be edited by the user, while remote models, which are accessed by HTTP, are used to integrate model libraries published by other contributors into the project. For example **Fig. 2** shows our case study configured with:

- Three local models named “DiscountVoyage”, “Data architecture”, and “Technical architecture”
- Three remote modes names “Application architecture”, “Businnes architecture”, and “Businnes entities”.
- Two locals libraries i.e. “PredefinedTypes 3.1.00” and “JDK 1.7.00”
- One remote library i.e. “Requirements”.

In practical terms, this means that once the project configuration has been established, the different models are viewed as a single model in the model browser as depicted in **Fig. 3**. They can be used transparently for modeling work.

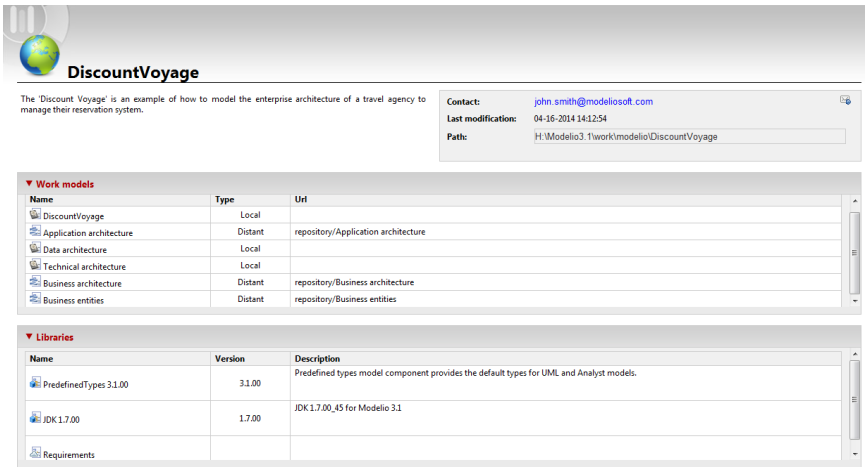


Fig. 2. Distributed framework configuration

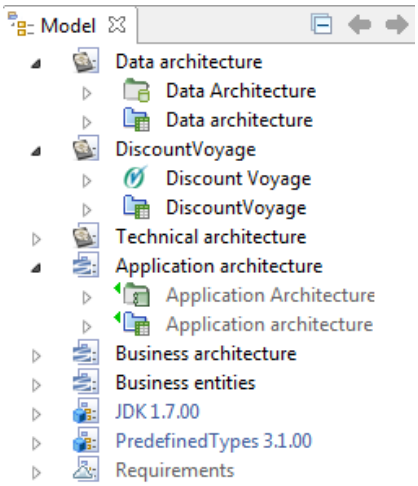


Fig. 3. Distributed framework rendering

The models combination used of each project depends of your set of concerns and of course theirs availabilities regarding the defined access right. This flexibility in term of model management allows the creation of different community or groups sharing set of models.

When you deal with accessibility configuration, you also have to deal with the object "absence". "Absent" objects are elements which are momentarily inaccessible but still known by their identifier and their type. In Modelio, absent objects are represented in simplified form (name, type icon) and therefore do not prevent the model from being used. Links modeled towards an "absent" object are retained. When an "absent" object is re-established and becomes accessible once again, these links are automatically re-established in their complete definition.

This mechanism enables, for example, a Modelio project open in the modeler to « survive » a network disconnection which deprives it of remote libraries and the elements they contain.

3.2 View, Viewpoint and lazy loading.

The SysML modelling standard defines a general-purpose modeling language for systems engineering applications, called the OMG Systems Modeling Language (OMG SysML™) [8]. Throughout the rest of the paper, the language will be referred to as SysML. SysML supports the specification, analysis, design, verification, and validation of a broad range of complex systems. These systems may include hardware, software, information, processes, personnel, and facilities.
The Open Group Architecture Framework, or TOGAF [7] modelling standard, is the de facto global standard for Enterprise Architecture. TOGAF provides the methods and tools for assisting in the acceptance, production, use, and maintenance of enterprise architecture. It is based on an iterative process model supported by best practices and a re-usable set of existing architecture assets.

Both these standards and others ones, define the view and viewpoint concepts. In short, a view is a portion of your system i.e. a representation of your whole system from the perspective of a related viewpoint. A viewpoint is a particular set of concerns. The implementation of these two concepts under Modelio allows, according to a specific set of concerns, a reduction of manipulated element. For example **Fig. 5** depicts the Modelio full viewpoint where many elements are shown and necessarily loaded in memory. Contrary to the Modelio trace viewpoint, cf.

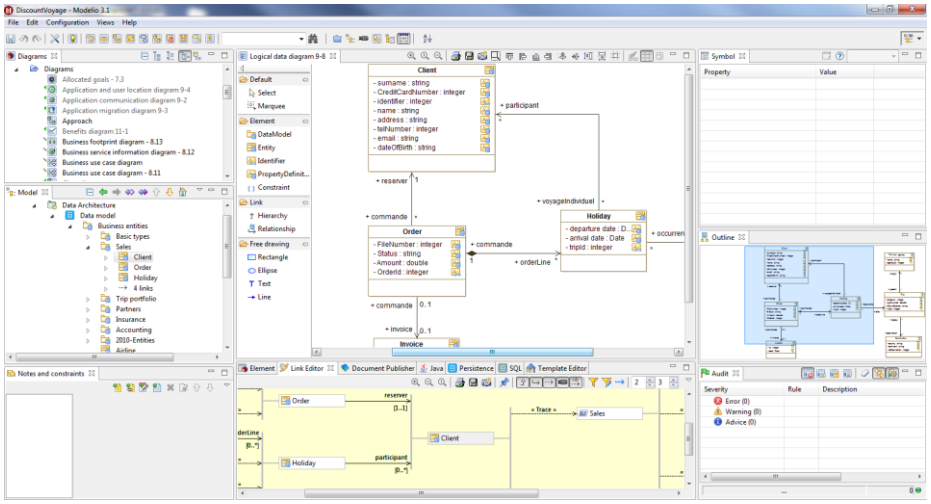


Fig. 4, focusses to the links from a given element but not to other relative concept. The association of this view/viewpoint mechanism and a lazy loading is able to minimize as much as possible the number of element loaded in memory and consequently the size of allocated memory.

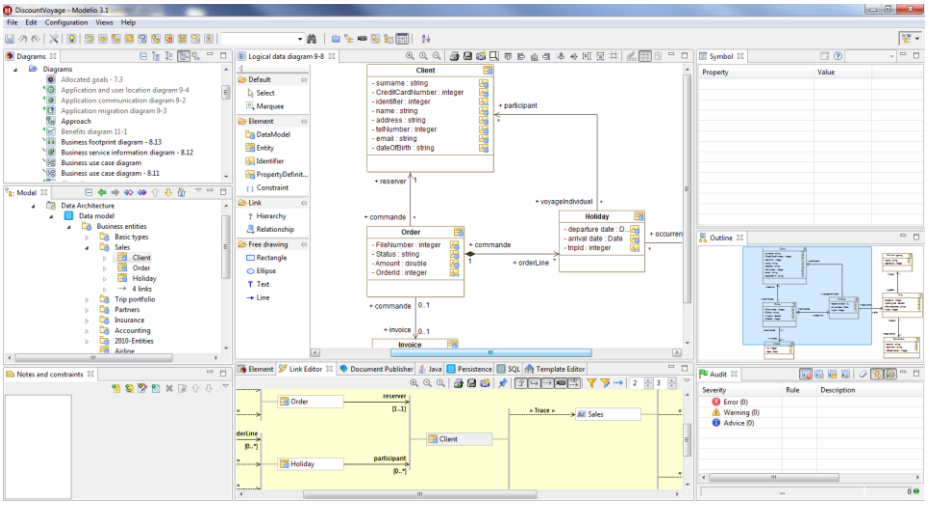


Fig. 4. Modelio full viewpoint

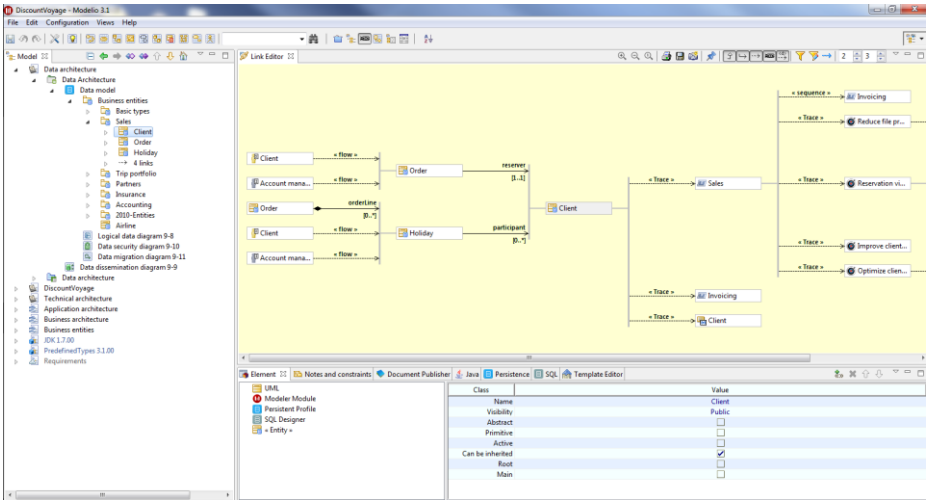


Fig. 5. Modelio trace viewpoint

4 Conclusion

As we have already stated the MONDO project is currently at its early stage so most optimization attempts must be specified, implemented and of course evaluated on relevant and industrial system design cases. But we are optimistic about expected results mainly because some techniques already present in current tools and presented in this paper have shown good results and will improve as the project advances further.

Acknowledgements

The research leading to these results is partially supported from the European Community Seventh Framework Programme (FP7/2007-2013) under grant agreement no FP7-611125. We would also like to acknowledge all the members of the MONDO (Scalable Modelling and Model Management on the Cloud) Consortium for their valuable help.

References

1. "Modelio," [Online]. Available: http://www.modeliosoft.com [Accessed 25

June 2014].

2. "MONDO Project Web Site" [Online]. Available: http://www.mondo-project.org/. [Accessed 25 June 2014].

3. Ujhelyi, Z., Bergmann, G., Hegedüs, Á., Horváth, Á., Izsó, B., Ráth, I., Szatmári, Z., and Varró, D., "EMF-IncQuery: An Integrated Development Environment for Live Model Queries", Science of Computer Programming, 2014.

4. Ujhelyi, Z., Horváth, Á., Varró, D., Csiszár, N I., Szőke, G., Vidács, L., and Ferenc, R., "Anti-pattern Detection with Model Queries: A Comparison of Approaches", IEEE CSMR-WCRE 2014 Software Evolution Week: IEEE, 02/2014.

5. Szárnyas, G., Izsó, B., Ráth, I., Harmath, D., Bergmann, G., and Varró, D., "IncQuery-D: A Distributed Incremental Model Query Framework in the Cloud", ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems, MODELS 2014, Valencia, Spain, Springer, 2014.

6. Dávid, I., Ráth, I., and Varró, D., "Streaming Model Transformations By Complex Event Processing", ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems, MODELS 2014, Valencia, Spain, Springer, 2014.

7. "TOGAF® Version 9.1 Enterprise Edition Web Site" [Online]. Available: http://www.opengroup.org/togaf/ [Accessed 25 June 2014].

8. "SysML Version 1.2 Web Site" [Online]. Available: http://www.omg.org/spec/SysML/1.2 [Accessed 25 June 2014].
9. Dimitrios S. Kolovos, Louis M. Rose, Nicholas Matragkas, Richard F. Paige, Esther Guerra, Jesús Sánchez Cuadrado, Juan De Lara, István Ráth, Dániel Varró, Massimo Tisi, and Jordi Cabot. 2013. A research roadmap towards achieving scalability in model driven engineering. In *Proceedings of the Workshop on Scalability in Model Driven Engineering* (BigMDE '13), Davide Di Ruscio, Dimitris S. Kolovos, and Nicholas Matragkas (Eds.). ACM, New York, NY, USA, , Article 2 , 10 pages. DOI=10.1145/2487766.2487768 http://doi.acm.org/10.1145/2487766.2487768

Flexible Model-Based Simulation as a System's Design Driver

Jean-Philippe SCHNEIDER[1], Eric SENN[2], Joël CHAMPEAU[1], Loïc LAGADEC[1]

[1] UMR 6285 Lab-STICC, ENSTA Bretagne, 2 rue Franois VERNY 29806 BREST CEDEX 9

[2] UMR 6285 Lab-STICC, Université Bretagne Sud, Rue de Saint-Maudé BP 92116 56321 LORIENT Cedex

Abstract. Complex systems traditionally involve partners from different companies with their own domains of expertise. During design stages, these partners need to exchange pieces of information and to debate around architectural and implementation choices.
Model Driven Engineering for System Engineering simplifies system knowledge sharing, while simulation provides sound results to drive debate. As a consequence, gaining a flexible and dynamic tool that models and simulates the architecture is highly valuable.
In this paper we focus on the functional architecture design and analysis steps of the system engineering process. We identify adaptation to existing system engineering process, tool modularity and interaction with models as three grounding principles for a flexible system model simulation tool. We show that meta-modeling and layered architecture for a simulator are enabling technologies for our three principles. We also demonstrate the use of these technologies by implementing a simulation tool in the context of a sea-floor observatory project.

1 Introduction

System engineering is an interdisciplinary activity during which experts from several domains having an holistic approach look for the near optimal system design to answer to a client's needs [6]. Interdisciplinary approach requires the ability to work with partners with different vocabulary and work techniques. Looking for a near optimal solution implies identifying and comparing multiple design solutions for the system. One of the risks in an interdisciplinary context is that each expert focuses on its domain of expertise and looks for a locally optimal solution (for its domain) that is not necessarily the globally optimal solution (for the system seen as a whole) [10]. Adopting an holistic approach reduces this risk. Holistic approach, interdisciplinarity and near optimal solution are linked problematics. The holistic view of the system must be shared amongst experts. Model-Based System Engineering (MBSE) is an approach in which system engineering artifacts such as the functional architecture are models [4]. Models are abstractions of the system and can be used to share knowledge

between experts [11]. Collaborative design gives all experts the opportunity to express their ideas. It is then possible to take into account the opposite point of views and to find more alternatives of system design. Simulation helps this collaboration by giving a concrete realization to the discussions [3].
The simulation tool used in the collaborative context should be fitted to work across domains of expertise. Each expert should find the main terms defined in its domain. For example, in a sea-floor observatory project, mechanic and software experts are involved. In such a project sensors are deployed underwater and tightness is a major requirement. This requirement has a translation in the mechanic domain and also in the software domain. Mechanically tightness means that the system has seals and is assembled in a manner that ensures that water will not enter into the system. In the software domain, tightness means that there is a way to measure the pressure into the system. In case of an abnormal evolution of pressure indicating a failure in tightness, a message is sent back to supervisors. Mechanical and software models should be put together to enable the simulation of the tightness function. The simulation outputs should be adapted to provide output meaningful for mechanics engineers and also for software engineers.
In this paper, we describe three principles that ground a flexible simulation tool:

- Adaptation to system engineering processes (current, and emergent/future);
- Extensibility of the tooling;
- Interaction with models.

Companies have already defined their system engineering process and use tools to support the process. Our approach is based on the respect of what has already been done in the companies. Multiple domains such as mechanics or software are involved in the design of a system. It is not possible to create a tool taking natively into account all of these domains. Domain experts should be able to add new functionality to the tool and remove those which are not required. Models are prototypes. Domain experts are using them to debate over the design alternatives for the system. They should be able to modify the models according to the result of their discussion. To realize our three principles, we describe a meta-model defining the concepts of a functional architecture. A functional architecture is the definition of the functions implemented in the system, their interactions and their structural layout. The meta-model describes the structure, the communication and the behavior aspects of functions that should be implemented by the system. In a Model-Based System Engineering approach the meta-model ensures independence from existing modeling tools. To obtain a flexible simulation tool, we define a layered and component-based architecture. Layers group tool functionalities according to their degree of specificity to a domain of expertise. Components isolate functionalities and provide a convenient way to reuse them.
The rest of the paper is organized as follows. Section 2 describes some related work. Section 3 first provides a motivating example for this work coming from a sea-floor observatory project we were involved in. Section 4 provides a descrip-

tion of the grounding principles. Section 5 describes the technologies usable to implement a flexible simulation tool.

2 Related Work

Different formalisms can be used to model a system functional architecture such as Enhanced Functional Flow Block Diagram (EFFBD) and SysML models. EFFBDs are an extension of Functional Flow Blocks Diagrams. Functional Flow Blocks Diagrams define functions and their sequence of execution. EFFBD adds data flow to the functional architecture modeling [12] and execution through Timed Petri Nets has been described in [16]. However, once the simulation is started it is not possible to have interaction with the running model.
SysML [17] enables to model a system architecture throughout the system design cycle. Functional architecture is modeled using Block Definition Diagrams for the structure and Activity Diagrams for the behavior. The link between structure and behavior is obtained through an allocation relationship. SysML models can be used as entry model for simulation tools [13] through model transformation. However, in SysML every structural definition (whether implementation or functional) relies on the concept of block. Using the same concepts at the functional and implementation level implies that designers should be careful not introduce implementation details into the functional architecture. The functional architecture should only describe what the system will do with no implementation choices so that multiple architectures are investigated to find the best one.
Two different approaches may be used to enable the simulation of models defined in a modeling tool: extend the modeling or simulation tool to introduce execution capabilities or use one tool to do the modeling, serialize the model in an interchange format which will be imported in the simulation tool.
Extending a modeling tool can be made using a plugin as suggested by Radjenovic and al. [15]. Their approach is structured in three steps with one design model, one simulation model and finally the simulation execution. The simulation tool is extended through a plugin enabling to extend the understanding of multiple input model formalism. However, this approach requires knowledge of the internal functioning of tooling and access through an API, which is not always possible with proprietary tools.
On the opposite, Karsai and al in [9] advocate using an interchange format between modeling and simulation tool. This approach relies on meta-modeling and model transformations. A model transformation is required to transform the architecture model into the interchange format and another one is required to transform the serialized architecture model into a simulation model. The interoperability of the tools relies on the interchange format. The interchange format may be custom as the goal is to provide a model exchange backbone for multiple tools in the same design process. We favored this approach as we do not want to modify existing tools which are known by designers. In our case, we have to adapt to the model serialization format of already used modeling tools.

3 Motivating Example

The MeDON [7] project aims at designing a proof of concept for a sea floor observatory in coastal areas. A sea-floor observatory is made of a set of underwater sensors and of computing servers. The scientific goal of the MeDON observatory is to passively detect sound sources in the area of Brest in France without defining their location. Hydrophones are deployed to acquire underwater sounds. Algorithms were defined to detect contributions to underwater sound above the mean sound level.
During the design phase of the MeDON project, experts from electronic, software and electronic fields among others were involved. Experts worked in their specific field of expertise and had interactions together only during progress study meetings each six month. There was no knowledge repository to store a common view of the observatory design. Decisions impacting every fields of expertise were made according to field specific goals without coordination with other experts. Some of these decisions had huge impacts on work already done. For example, a deployment site was chosen at the beginning of the project. However, this choice had to be modified due to energy supply shortage. This choice was necessary but it implied rework on the deployment of the data computing softwares and on the choice of the servers. With a functional architecture independent from any technology, it would have been possible to reuse a lot of engineering work. Besides, a better communication between experts about the possibility of power supply shortage would have led to the definition of at least two alternatives architectures. So, when the change occurred, the switch of architecture would have been anticipated.
The MeDON project is now being upgraded to include the localization of the sound sources. Learning from our experience, we decide to use a SysML-based system architecture model to define the system architecture and to have a shared view on the system. An algorithm based on the difference of sound arrival time between each hydrophone has been selected to locate sound sources. The sound is acquired by at least three hydrophones in a two dimensional approximation of space. One of the hydrophones is chosen as reference. Each hydrophone acquire sound signals. The acquired sound is then analyzed to detect the presence of a signal higher than the noise. If one is found it is considered as a detection. For each detection on each hydrophone a difference is made between the time of reception on the hydrophone and on the reference. It is then possible to have a location of the sound source [18]. We model a functional decomposition of the algorithm with SysML. The Figure 1 shows the Block Definition Diagram we obtained. A Passive Acoustic Monitoring system *PAMSystem* is made of *Acquisition* and *Computing* functions. The *Acquistion* must contain a *RawAcquisition* function to acquire the raw signal. The *Computing* function must contain a *Localization* function which perform the localization. The *Detection* function analyzes the signal from *RawAcquisition* to check for a value higher than the mean signal value. This function can be grouped either into the *Acquisition* function or the *Computing* function. This is an architecture alternative that should be investigated. We instantiated the architecture in which the *Detection* function

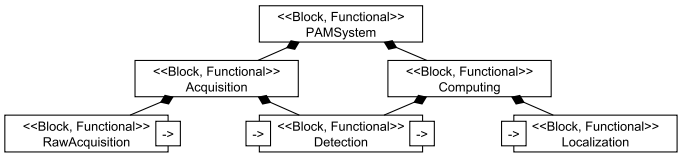


Fig. 1. Block Definition Diagram of a Passive Acoustic Monitoring System

is grouped into the *Acquisition* function. The result is shown Figure 2. In this

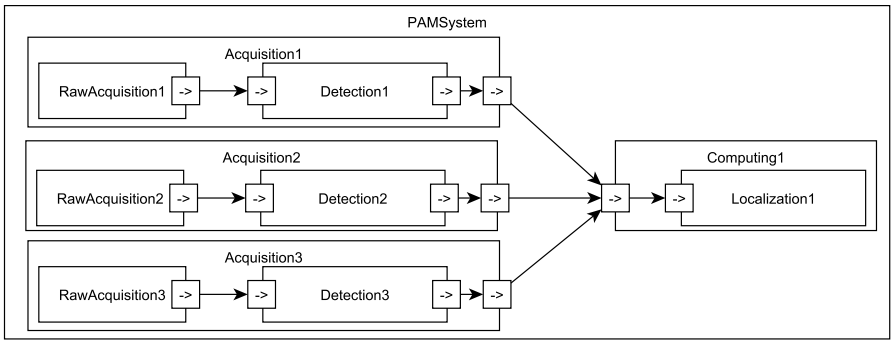


Fig. 2. Internal Block Definition of a Passive Acoustic Monitoring System

work we adopted an approach mixing data scientists' point of view for the block definition and the point of view of software engineers for the data flow. However, this is not enough. We must take into account the point of view of experts from the other domains involved in a sea-floor observatory such as electrical experts. Besides, the tooling is not dynamic and flexible enough to enable users to model and simulate alternative of architecture.

4 Grounding Principles for a flexible simulation tool

In this section, we will introduce three principles underlying the development of our simulation tool. First, we know that industrial companies have well defined system engineering processes. So we think that our tool has to adapt to these processes and not the other way round. Second, we think that the users' needs will continuously evolve so the simulation tool must follow these evolutions. As a consequence, the simulation tool must be extensible. Third, we would like that the model and the simulation become an active support for reflections. This leads to interactions with models.

4.1 Adaptation to Deployed System Engineering Process

The ISO 15288 standard [1] defines the activities performed during the system engineering process. We draw our interest on the functional architecture design step. The functional architecture describe the functions implemented in a system. These functions come from the requirement analysis made with the users. The functions defined through the requirements analysis will be organized in several alternatives of functional architectures. Definition and comparison of functional architecture alternatives are essential as the choices made at this step of the process drive the whole system realization. Simulation is one tool to perform this comparison.
Simulation relies on the modeling activity. Models define the abstraction level of the simulation and serve as entry point. A model driven approach for simulation is made of four steps [2]:

1. Conceptual Modeling: definition of the system model at a given abstraction level.
2. Tool independent simulation modeling: translation of the previous model into a simulation formalism such as Discrete Event. The independence from simulation tool enable reuse of the model.
3. Tool specific simulation model: translation of the previous model into a model using the concepts defined in the chosen simulation tool.
4. Implementation.

In our case, the functional architecture can already be modeled using languages such as SysML [5]. These languages are already used in companies' system engineering process. In order to be adopted, a tool must comply with industrial processes [8]. The modeling of the functional architecture is equivalent to the conceptual modeling step. It is made with existing tools so that we comply with companies' process and tools. As a result, the simulation tool is decoupled from the functional architecture modeling tool while sticking to the system engineering process. A meta-model is provided to describe the elements of the functional architecture to simulate. Intermediate models compliant with the defined meta-model are independent from any simulation tool. The intermediate models can be obtained through model transformations from system engineering tool knowing their meta-models.

4.2 Adaptation to User's Needs

Complex system design requires work from experts coming from multiple domains. Each expert has its own view on the system through its domain vocabulary and also on the metrics given by the simulation. The simulation tool must be able to take into account the differences between domains. Experts need a tool adjusted to the current situation they are facing. Experts should be able to add new functionalities to the simulation tool and remove the ones they are not using.
This requires a modular approach like the one used in the Linux Kernel. The

Linux kernel is made of a set of modules. Each module has an unique purpose such as handling a USB device or printing. A module can be loaded and unloaded. However, some modules are essential to the stability of the system and they can not be unloaded.
Like the Linux kernel, the simulation tool should be made of modules that can be loaded and unloaded by domain experts as required by their current task. Besides, a distinction between modules should be made. Some modules are at the basis of the simulation and so should not be unloaded. Other modules deal with domain specific activities and may be loaded and unloaded at will.

4.3 Interaction with Functional Architecture Models

Interaction with models consists in observing the simulation and its results and in modifying the simulated model. These activities serve the purpose of:

- Enabling to define new alternatives of functional architecture by debating;
- Simulating the different alternatives;
- Comparing the results of the simulation.

One of the goal of functional architecture is to define groupings of functions. In the simulated models the functions groups should be modifiable to perform tests on different alternatives. A function can be seen as an assembly of basic operations. The order in which the basic operations are performed or their nature should be modifiable. Those interactions with models are risky: deadlocks can be created by the modifications. Furthermore, modifications may cause a loss of coherency between the simulated model and the simulation results. The simulation results must be linked to the simulated model. So the simulation engine must take into account all the modifications performed on the model so that the simulation results are still valid.
Tests on the structure of the model should be performed to avoid these risks. At runtime, a deadlock check should also occur. The modeled elements themselves should also provide pieces of information about which modifications users are allowed to perform on them. Unauthorized modifications should be blocked by the model elements.

5 Enabling technologies for the Guiding Principles

In this section, we will introduce the technologies used to implement the three principles. Meta-modeling and model transformation implements the adaptation to system engineering process. Layered architecture helps to implement tool extensibility.

5.1 Meta-Modeling and Model Transformations

We defined a meta-model describing functional architecture models. This meta-model shown Figure 3 is based on the function description in [14]. Functions are

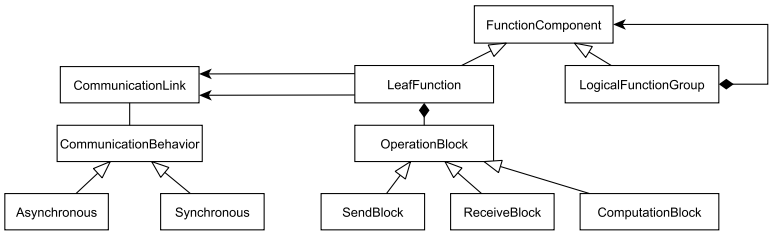


Fig. 3. Meta-model defining a functional architecture

seen as actions performed by a system and are allocated to constituent of the system. We extracted the definition of the functions and elaborate on it. Our meta-model is made of the definition of the functional structure, the communication between functions and the behavior of functions.

The functional structure is described by a Composite pattern between the *FunctionComponent*, *LeafFunction* and *LogicalFunctionGroup* meta-classes. The meta-class *LeafFunction* describes the basic functions of the system. A logical function group (LFG) can be made of other LFGs and of basic functions. in the example of our sea-floor observatory, we have a LFG called Acquisition that groups two *LeafFunction*s *RawAcquisition* and *Detection*. The two latter are *LeafFunction*s as they are not further broken down.

Basic functions can communicate through communication links. Each communication link has a behavior to define if the communication will be synchronous or asynchronous. To bring flexibility in modeling the behavior of functions, we reified the concept of communication link. We decouple the behavior of the communication from the behavior of the function. Changes in the function behavior will not affect the way communications are performed. *LeafFunction*s know each link through an alias. There is one link per exchange between two functions. This mechanism is similar to ports in a component-based modeling. Unlike ports, our modeling of links do not allow broadcast. However, it eases the analyses of the exchanges between functions as exchanges between a sender and multiple receivers must be explicitly modeled.

The behavior of a function can be described as a sequential list of basic operations such as sending or receiving data and performing a computation. The sending operation is described by the name of the communication link on which the data are sent. The receiving operation is described by the name of the communication link from which the data are read. The computation operation is described by the computation duration.

To adapt ourselves to the process and tools used in the industry, the meta-model define an intermediate representation of functional architecture. The functional architecture is modeled using companies' internal modeling tool. Model transformations extract the data relevant to functional architecture from companies' model and create a new model compliant with our meta-model. This new model is used to define the simulation to perform independently from the companies

tools used for modeling the functional architecture at the beginning.
Our meta-model was designed in the context of the sea floor observatory example and uses vocabulary of data processing such as *Computation*. However, it can be easily extended to suit more generic needs. An *Action* meta-class can be created. The *Computation* meta-class will inherit from it. The class *CommunicationBehavior* can be renamed in *LinkBehavior*. Classes detailing the behavior of flows inheriting from *LinkBehavior* can be created.

5.2 Layered Architecture

To obtain tool extensibility we rely on a layered architecture each layer being made of software components. In a layered architecture each layer uses services from the lower level layer and provides services to the upper level one. It is then possible to build a new service providing business oriented data built from tool services. Components have a well-defined interface. Their functional responsibilities are clearly identified. It is possible to switch two components with the same data inputs and outputs. Components may also have the ability to be loaded at runtime. New functionalities can then be added to the tool at runtime.
Using a layered and component-based architecture has several advantages. First, using component enable to co-locate pieces of code having the same role. When a modification is required, locating the area in the code that must be modified is easy. Second, using layers and components requires to clearly identify the interfaces between the components. This enables to write new components and integrate them in the simulator. The only condition is to comply with the interfaces. However, using layers and components for the architecture have some disadvantages. The complexity of the architecture of the simulator may be increased. Information useful in a component may follow a complex path before gaining the targeted component.
We decompose the simulation tool into three layers as shown Figure 4. The *Core*

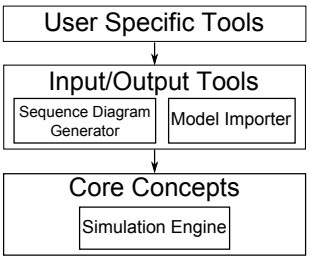


Fig. 4. Decomposition in layers of the tool

Concepts layer contains the simulation engine component. This component contains the implementation of the meta-model.
The functional architecture is modeled outside our simulation tool and model

transformations are performed to import it in our tool. A model importer component should be responsible for performing the model transformation. Besides, the simulation should generate output rendered into an user friendly format. An export module per output format should be implemented. All those components are located into the *Input/Output Tools* layer on top of the *Core Concepts* layer. The *Company Specific Tools* layer will contain the different components developed specifically for the users' needs. For example, a component may be written to compute the global waiting time the different functions and display the obtained value.
An example of component decomposition is the integration of a sequence diagram utilities. Sequence diagrams enable to visualize message exchanges between functions. We defined two components: one that displays directly a sequence diagram and one which creates a log file using a format readable by an external tool in such as the sdedit tool[3]. A component approach is well suited because in both case information about the message exchanges are the same i.e. the identity of the sender, the message name, and the identity of the receiver. Besides, the behaviors of both component are really different, the first manage a display and the second one write data in a file. The link between the execution engine and the components is made through an interface detailing the data structure exchanged between simulator components.

6 Conclusion

In this paper we presented the three necessary principles to obtain a flexible simulation tool at the functional level. First, we think that such a tool should adapt to the system engineering processes already in use in companies and not modify it. Second, we advocate for a tooling able to be adapted to specific needs. Third, the simulation tool must enable to play with the models simulated. We also gave a list of enabling technologies. Meta-modeling and model transformations may support the adaptation to existing system engineering process. Flexibility is achieved through the independence from deployed tooling. Module-based tooling enable the adaptation of the tool to the user needs. Flexibility is achieved by the ability to load and unload modules at will according to the project environment.
A future work is to extend our functional architecture metamodel. We want to reify the composition link between the *LogicalFunctionsGroup* and the *FunctionComponent* metaclasses. It will then be possible to add pieces of information to specify the link such as an alternative identifier. This identifier will ease the process of implementing, simulating and comparing functional architecture alternatives.

Acknowledgments

This work has been done with the financial support of the French Délégation Générale de l'Armement and of the Région Bretagne.

[3] http://sdedit.sourceforge.net/

The authors also wish to thank Zoé Drey and Ciprian Teodorov for their valuable input to this paper.

References

1. Arnold, S.: Iso 15288 systems engineering system life cycle processes. International Standards Organisation (2002)
2. Cetinkaya, D., Verbraeck, A., Seck, M.D.: Model transformation from bpmn to devs in the mdd4ms framework. In: Proceedings of the 2012 Symposium on Theory of Modeling and Simulation-DEVS Integrative M&S Symposium, p. 28. Society for Computer Simulation International (2012)
3. D'Aquino, P., Le Page, C., Bousquet, F., Bah, A.: Using self-designed role-playing games and a multi-agent system to empower a local decision-making process for land use management: The selfcormas experiment in senegal. Journal of artificial societies and social simulation **6**(3) (2003)
4. Estefan, J.A., et al.: Survey of model-based systems engineering (mbse) methodologies. California Institute of Technology, Pasadena, California, USA May **25** (2007)
5. Friedenthal, S., Moore, A., Steiner, R.: A practical guide to SysML: the systems modeling language. Elsevier (2011)
6. Haskins, C., Forsberg, K., Krueger, M., Walden, D., Hamelin, R.D.: Systems engineering handbook. INCOSE. Version **3.2** (2010)
7. Interreg IVA: Marine edata observatory network (2013). http://medon.info/
8. Kapurch, S.J.: NASA Systems Engineering Handbook. DIANE Publishing (2010)
9. Karsai, G., Lang, A., Neema, S.: Design patterns for open tool integration. Software & Systems Modeling **4**(2), 157–170 (2005)
10. Klein, M., Sayama, H., Faratin, P., Bar-Yam, Y.: The dynamics of collaborative design: insights from complex systems and negotiation research. Concurrent Engineering **11**(3), 201–209 (2003)
11. de Lange, D., Guo, J., de Koning, H.P.: Applicability of sysml to the early definition phase of space missions in a concurrent environment. In: Complex Systems Design & Management, pp. 173–185. Springer (2012)
12. Long, J.: Relationships between common graphical representations in system engineering. Vitech white paper, Vitech Corporation, Vienna, VA (2002)
13. McGinnis, L., Ustun, V.: A simple example of sysml-driven simulation. In: Simulation Conference (WSC), Proceedings of the 2009 Winter, pp. 1703–1710. IEEE (2009)
14. Pfister, F., Chapurlat, V., Huchard, M., Nebut, C., et al.: A design pattern meta model for systems engineering. 18th International Federation of Automatic Control (IFAC 2011) (2011)
15. Radjenovic, A., Paige, R.F., Rose, L.M., Woodcock, J., King, S.: A plug-in based approach for uml model simulation. In: Modelling Foundations and Applications, pp. 328–339. Springer (2012)
16. Seidner, C.: Vérification des effbds: model checking en ingénierie système. Ph.D. thesis, Nantes (2009)
17. SysML, O.: Omg systems modeling language (2013)
18. Zimmer, W.M.: Passive acoustic monitoring of cetaceans. Cambridge University Press (2011)

Putting Real Production Software in the Loop, Methodologies Enabling SW Co-Development Between OEMs and Tier 1s

David Bailey, Guillaume Francois and Gregory Nice

ETAS GmbH Borsigstrasse 14, 70469, Stuttgart, Germany
David.bailey@etas.com, Guillaume.Francois@etas.com

Abstract. With software gaining importance as the main contributor both to functionality and differentiation in the automotive market place and its relevance to quality, safety and customer satisfaction, its place in the development process and the methods available to ensure short development cycles and a simultaneously high level of quality are coming under strain. Both model-based and abstracted code – not specific to the final production target, are in use in the earlier phases but these often do not provide code which is testable in a meaningful way for the final product. In this paper we will explore methodologies which allow target independent code to be produced and managed as a product within the development process – establishing clear linkage between development code and the final product and accountability and traceability throughout the process. We will leverage the increasing implementation of Autosar and proliferation of model based sw development techniques in the process.

1 Introduction

The past ten years has seen an exponential growth in the amount of software going into vehicles both to perform and improve core functionality, such as fueling, combustion control, valve actuation and breaking as well as to extend the capability of the vehicle with advanced driver assistance, connections to roadside, service bay and ubiquitous wireless infrastructure bringing the Internet in ever-closer connection with the Automotive control system. At the same time the era of "Mass Customisation" has bred a generation of consumers who increasingly look, not only to purchase the latest and best technologies but who also expect to be able to configure products in the way that suits them and their life-style. This has driven an exponential growth in product variants and as a consequence software variants that require managing throughout the product life-cycle. One would think that this would be sufficient to present challenges to an industry which has adopted electronic control as recently as the 1980's but on top of these challenges customers expect ever higher levels of safety and concern for the environment. This has also lead governments to increasingly

legislate in the areas of safety, emissions and sustainability in developed markets adding to the cost-burden on Western OEMS to get their products into markets. Whilst there is no doubt that high and ultra-high end technology is coming to the market in the shape of Teslas, Bugattis, and other high-end products, there is also a diametrically opposed trend occurring in parallel for simplicity and functionality both with the requirement to address the needs of the first-time vehicle purchaser in India or China and with a new generation of consumers in the West where the vehicle no-longer represents a status symbol but rather is regarded as a method for getting from A-B with as little fuss as possible, preferably with access to Facebook and Twitter.

To address all of these demands simultaneously is the reason why the Automotive industry must embrace a paradigm shift in its thinking, costing and planning for product development. The differentiators in the market are increasingly software based as are the potential risks of damage to brand and future viability for firms if Software is not managed with the same regard for safety and quality as any other component in the vehicle. In addition the OEM is increasingly playing the roll of systems integrator not only on the vehicle level but also at the level of individual ECU. Integrating software developed, in house, by a tier 1 and by 3rd party specialized suppliers in one mechatronic system. This fact also drives a demand for a consistency of architecture, interfaces and lifecycle management to ensure quality and traceability in the entire process

2 From V-Cycle to Model Based to Agile.

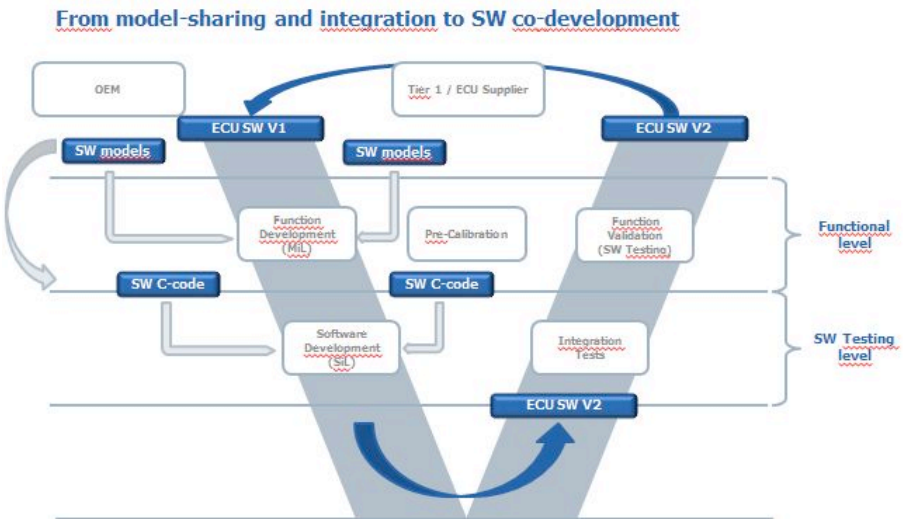


Fig. 1. From the V-Cycle to "Agile"

Considering both the driver for time to market and the requirement to manage, integrate and test software components to the required quality levels has necessitated both Tier 1s and OEMs to restructure their development processes in significant ways. From a high level requirement a feature or function within the system is typically modelled. Since there is no universal modelling tool – this typically involves a range of modelling tools in complete system, ranging from specialized environments such as Ricardo WAVE or GT-Power from Gama Technologies, to Simulink to C-code. There then follows an iterative development process around the function where the model function is refined until such time as the functional developer is satisfied that the desired functionality is reliably performed. At this stage it is not a given that the Functional model is perfect – rather that it appears perfect in the environment under which it is tested. Therefore it is essential to manage within this phase; model variants, configurations, tool versions and data-sets so that the finalized functional model can be reproducibly brought back into a further iteration step if an issue is found during the further development and validation stages.

Here we are already at a level where most OEMs and Tier 1s widely collaborate in the software development process. Namely sharing IP and functional specification on a model level. A desired function is developed by an OEM. It is tested in a Model-in-the-loop-environment with the associated plant, behavioural and environmental models. The Functional Model is then provided to the Tier 1 for further integration testing and is either converted into a further, more suitable form for model-based code generation directly or hand-coded in C before integration within the software build process for a specific ECU and target processor.

The challenges here for process integrity and traceability are significant.

Firstly the range and variety of models, tools, versions, configurations and associated data-sets is both broad and numerous.

Secondly the tested function model – which acts as a specification for the Tier 1 omits information and configuration steps with the rest of the software system and potentially also the mechatronic system that may have an effect on the specified feature “in-situ” within the overall system. Of these, OS behavior, interaction with other SW components within the system, for example diagnostic modules, execution order and quantisation effects are frequently areas where discrepancies between the “perfect” behaviour in the function is followed by implementations of the series-intent code which end up behaving differently in the target.

This fact has driven both the drive for common ECU SW architectures such as Genivi and more importantly for control systems, AUTOSAR and a range of tooling that enables pre-Autosar architecture to be tested in a common environment that can be shared between both OEM and Tier 1 that takes account as many of the areas for potential divergence in functional behavior between the target and the pure model-level environment.

3 The Virtual and Rapid Prototyping Environments

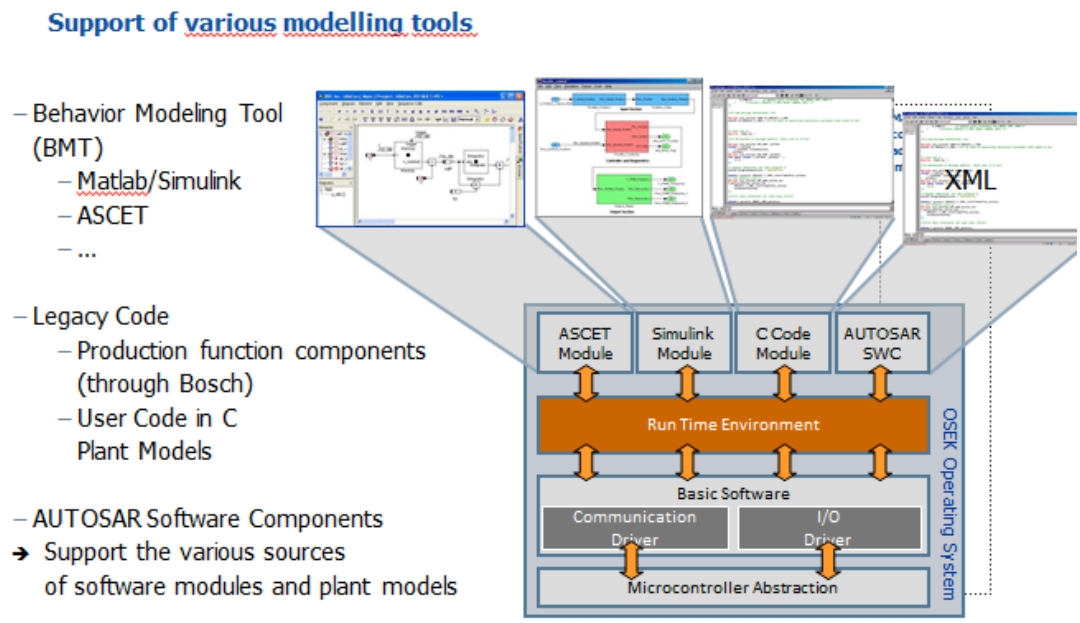


Fig. 2. A Virtual and Rapid Prototyping Environment

The requirements that therefore arise out of 1) The desire to develop in an agile way to improve quality and time to market and 2) The requirement to share and validate implementations of the functional specification in a meaningful way, lead to the requirement and productive use of Virtual and Rapid Prototyping environments at both Tier 1 & OEM. For these a number of essential features apply.

- Functional, behavioural, plant and environmental models need to be integrated in a common environment.
- This environment needs to provide features which allow the testing of changes to configurations in the OS configuration and other sw modules within the system in a manner which is as close as possible to the behavior which will be seen in the final target.
- A seamless transition between the prototype code and a real control system is also highly desirable to enable a functional test within a Real-Time active control system and potentially a base target ECU where the ECU code itself is not complete or not yet modified for the intended use case. In this case the environment should also support compiling to code for computing modules which process the function in parallel to the existing code

running on the ECU by means of software hooks and bypasses which allow a first – low effort integration of the new sw function into the control systems without going through a complete ECU software build.

Virtual Prototyping	**Rapid Prototyping**
Non real-time: Runs as fast as possible or with time scale. No connection to the real world. No I/O No communication buses	Meets hard real-time conditions
Stimuli or plant model required	Interacts with the real world. Comprehensive support for peripherals, analog and digital I/O & communication buses
Used for early validation and pre-calibration on the Windows® PC on the developer's desk	Validation and calibration on the test bench or on the road

Fig. 3 Virtual vs Rapid Prototyping

4. Typical Process Steps in SW Sharing Projects between a Tier 1 and an OEM

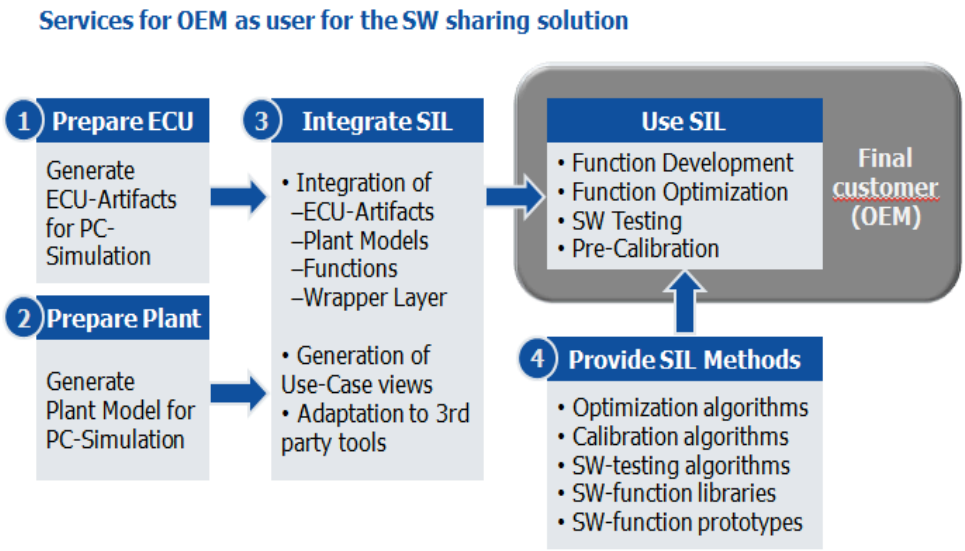


Fig. 4 Typical Process Steps in a Project employing software sharing between an OEM & Tier 1

Step 1 in the process usually involves the Tier 1. In most cases the Tier 1 will have existing tooling and methodologies for allowing their Platform software and Application software to run on a PC. In this particular case the Tier 1 uses the same OSEK OS or Autosar OS and RTE and compiles it for the PC target. If the aim is to test primarily on a functional level then software modules or groups of software modules (Functional Components or "FC"s) are compiled as .dlls capable of being integrated within the virtual prototyping environment via an Autosar RTE or an "RTE like" runtime environment that also supports legacy APIs and API calls. This is usually the role of the Tier 1 or the party which has the most responsibility in the project for SW development and integration. Again the ECU artifacts and the PC runnables need to be mapped and managed to ensure, traceability, repeatability and accountability. Frequently software components need to be stubbed to provide and interface to models or other stimuli instead of an interface to genuine HW.

Step 2. Is to prepare the plant. This involves deciding and selecting the required models and stimuli required within the environment to be able to test the SuT with the required test coverage and depth. The required data-sets, parameterization, models and other stimuli also need to be managed with the same regard as the software components mentioned above. This task is usually carried out by the party responsible for

the validation environment, frequently a 3rd party with a close working relationship with both Tier 1 and OEM. However both Tier 1 and OEM can frequently assume this role entirely or partly according to where the domain knowledge for the systems being modelled in the environment lie.

Step 3. Is to integrate the environment. This entails combining the simulation and the system under test within the test environment – mapping models and other stimuli to software component interfaces and integrating other tooling required for the test procedure – for example, calibration tooling, version and configuration management tooling, bus-analysis tooling and test automation tooling. This usually lies within the competency of the party responsible for the environment, in most cases a 3rd party.

Step 4 Is to define and provide the required test strategies and methodologies. These are usually provided by a 3rd party with input as required from both tier 1 and OEM

Step 5. The operational use of the SIL system is usually and OEM activity – nevertheless as with other test activities it can frequently be completely outsourced to a 3rd Party with the OEM or Tier 1 customer providing simply a list of requirements to be tested and the Pass/fail criteria. The system itself may also be employed at different departments within one OEM with various aims and work-splits. For example for pre-calibration or use within the OEMs own functional development process where the SIL environment provides a high-fidelity replication of the system as it will operate in the final series target.

5 Bringing Autosar and Agile Together

So far we have only considered the possibilities widely in use today based on the predominant sw architectures in series production at OEMs. Today and especially in the realms of Powertrain, ADAS, Chassis and Drivetrain are predominantly based on a software architecture proprietary to the Tier 1 or to the OEM. Whilst many Tier 1s offer an “Autosar Sandbox” for their OEM customers which allow easy integration with their legacy software – the vast majority of the code in these areas is legacy. However the move to Autosar is happening and happening rapidly even in these areas and at the very minimum a very good interface is required between the legacy architecture and Autosar in order to allow both migration of functions between architectures and reliable operation in the series target where a mixed architecture is employed. As far as the SIL environment is concerned this step towards Autosar offers a range of new possibilities that were only achievable before with a level of effort prohibitory to implementation in the past.

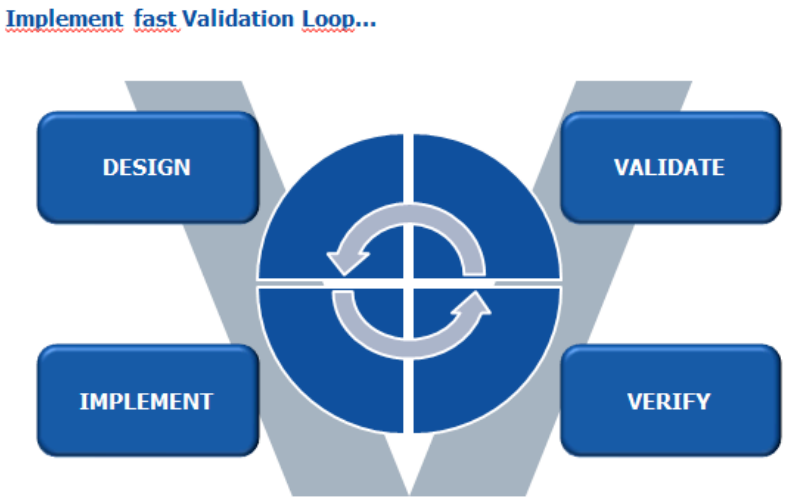


Fig. 5 Agile in Automotive is enabled by Autosar

The first major benefit brought by the shift to Autosar is that most Automotive Silicon today is delivered with a HW abstraction layer that supports the Autosar standard (ie an Autosar MCAL). As this has been so for a number of years a large amount of the code, Autosar or not, is already using a target independent API for communication with the microprocessor. This has significantly reduced the amount of stubbing required to enable code to be portable to the PC and also increased the portability of projects between processor platforms.

The RTE is non-target specific anyway, so any software in the system using the RTE for communication can be compiled 1 to 1 either for a real target or for the PC. Assuming there is a port of the OS for the PC available which allows for the same degree of configuration as on the target and an API or other communication mechanism by which other (non Autosar) Software components and elements within the environment (models, stimuli, test tools) then it is already significantly less, if not low effort to provide a build chain which is capable of taking the ECU artifacts intended for the series target and compiling these 1 to 1 for a PC based virtual validations platform. This, to the extent that it is not only possible off-the-shelf with full Autosar architectures but also extremely cost effective even with significant legacy content due to the fact that the system can be run and tested and developed against at very high fidelity, in Real Time or faster than real time. This has the corollary that many of the test activties that previously required real ECU HW, real engines and real vehicles to carry out can now be contemplated on the desktop. Thus offering a truly huge potential to cut time, effort and corresponding costs from the entire development process.

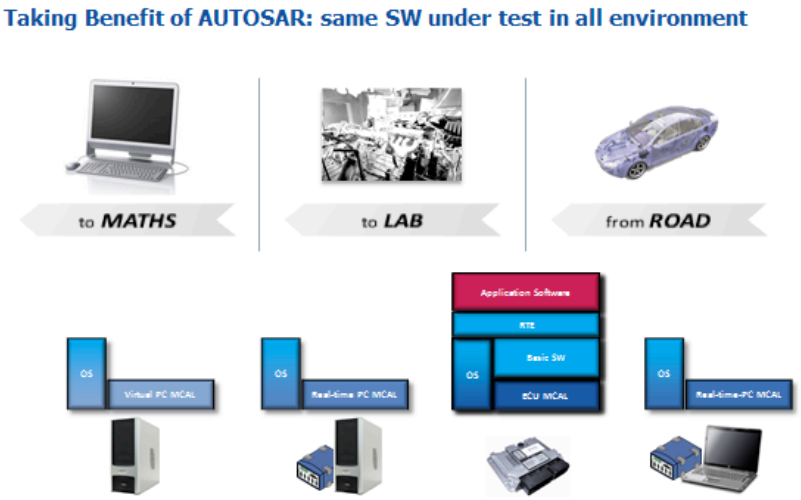


Fig. 6 The same SW can be compiled for any target at any stage of development

Aside from the obvious benefits of being able to front-load the development process, a number of other advantages emerge, some esoteric and some banal but all bringing a degree of much needed rationalisation to the exponentially growing demands within the industry. On the banal side – testing and development resources are spread throughout the world – fortunately or unfortunately there is no truly global market established so customs officials, especially in some of the lower-wage economies where development and testing is typically carried out, still have an interest to arrest, inspect and delay development and prototype ECUs and components. On average delivery time between a development centre is Germany and India ranges from 1-3 months. On the more esoteric side, once the control system is available on the PC steps to integrate this with Android, Genivi or other OS in a multicore envirnoment to address connected vehicle development becomes much more simple. Most crucially writing, testing and proving that code is safe has become key, since the systems we drive our children around in are controlled by the software that we produce.

If we consider some of the issues arising our of Philip Koopman and Michael Barrs analysis of Toyota's software presented in the case "Bookout & Schwarz v Toyota", most of the issues raised concern fundamental flaws in design and architecture that can only be address at the deeply embedded level. Therefore there is an increasing demand that the virtual validation methods we use are as close to reality as possible. In todays model-based world, which is largely regarded as state of the art

there is increasing testing on that level but without the ability to dive down into the behaviour at the lower levels and here there are plenty of banana-skins lurking.

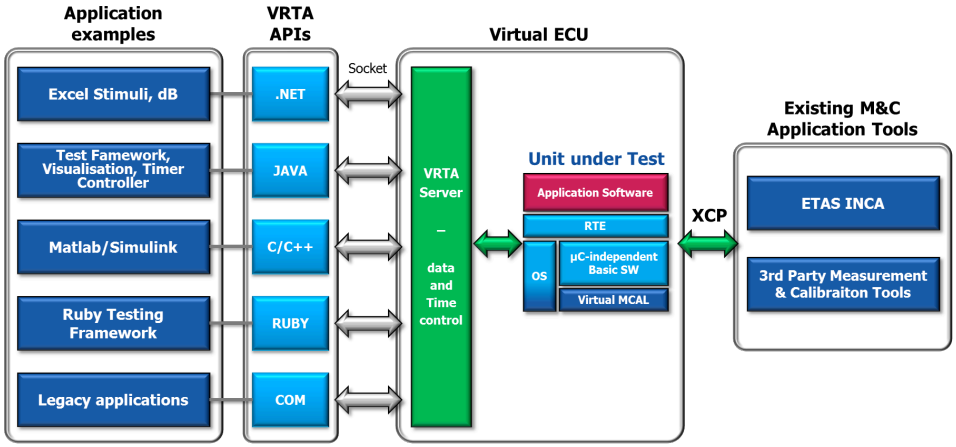


Fig. 7 APIs offering access to all levels of the sw architecture from external test tooling and simulation enviroments

6 Conclusion

The number, complexity and scope of embedded control systems in the vehicle is increasing exponentially. These systems now perform the key functions of the vehicle so software has to perform on all levels – primarily of course with regard to safety, but also with respect to fulfilling customer requirements with regard to features, differentiation and reliability. To put this in context a 737 Airliner contains 6,5 Million lines of code. The latest S-class from Daimler contains over 20 million. In terms of driving or flying hours the average platform racks up 1000x more flying hours than the entire fleet of Boeing 737s has flown since 1968. Whilst it must be conceded that not all of these systems are safety critical – the importance of introducing rapid iteration cycles and traceability throughout the development process cannot be underestimated. This can only be achieved with a paradigm shift in the industry with regard to the way software is regarded, developed and managed – with the same importance or more than any other part of the vehicle. If the automotive industry does not step up to the challenge then increasingly it is looking as if others will do so in its stead.

References

1. http://www.safetyresearch.net/blog/articles/toyota-unintended-acceleration-and-big-bowl-%E2%80%9Cspaghetti%E2%80%9D-code

2. http://www.safetyresearch.net/Library/BarrSlides_FINAL_SCRUBBED.pdf

3. Pavey & Winsborrow, "Demonstrating Equivalence of Source Code and PROM Contents", Computer Journal Vol 36, No 7, 1993

4. Charette, "This car runs on code", IEEE Spectrum, Feb 2009

System Engineering, for a Cognitive Sciences Approach

Thomas Peugeot, CSEP

Airbus Defence & Space
Metapole,
1, Boulevard Jean Moulin, 78990 Elancourt Cedex

Abstract. This article proposes to throw some light on the strong relationship between Cognitive Science and System Engineering. In the same manner that Herbert Simon justified hierarchical organizations with the argument that individuals had "bounded rationality", this article tries to show that most of our System Engineering processes are techniques to compensate for our cognitive boundaries. Practical implications for SE justification and SE education are discussed.

1 Introduction

Be prepared for the truth: despite 6 million years of fast paced growth, our brain is still a limited engine. The brain cannot hold more than 50 000 patterns; the brain cannot ingest more than 15 concepts per days; the brain cannot manipulate more than 7 ideas simultaneously and the brain cannot stop from flying tactically to the first solution instead of first considering strategically all available venues.

Those 4 cognitive rules were proposed by Alan Newell in "Unified theories of cognition" (UTC, ref. 1). This article proposes to explain some System Engineering process with Newell's rules. In the same manner that Herbert Simon justified hierarchical organizations with the argument that individuals had "bounded rationality", this article tries to show that most of our System Engineering processes are best practices both to compensate for our individual cognitive boundaries and to scale, with the number of talents involved, our ability to develop systems.

2 Unified Theories of Cognition

UTC was published by Newell in 1991 (notice that Newell was knowledgeable about system engineering since he was a fellow of Herbert Simon, see ref 9). In UTC, Newell proposes a fruitful theory of cognition. For instance, at Airbus Defence & Space, our human behavior simulations are influenced by Newell's models (see ref 4 for an example).

Here, we are not interested in Newell's response functions. We are interested in the 4 rules that Newell reminds at the beginning of the book. Those rules might not be aligned with the latest cognitive sciences theories but they are enough for our argument.

Max processing

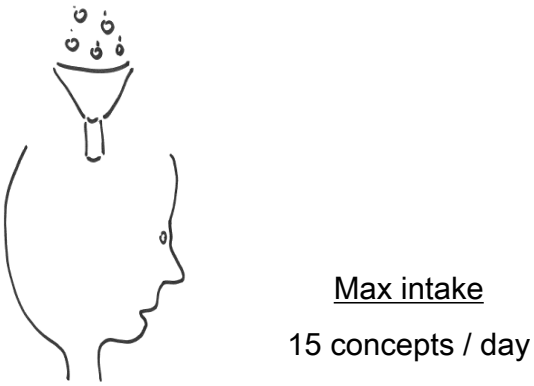
7 concepts

Rule 1: “7 +/- 2”. This rule is the most well known. 7 +/- 2 is the amount of concepts that can be manipulated simultaneously by an individual. For instance, it is known that a map should not have more than 7 different colors.

Max storage

50 000 concepts

Rule 2: 50 000 concepts per individuals is the amount of concepts an individual has in his brain store. For instance, when you ask a software engineer a question about his program, it is a rule of thumb that he can answer immediately if his program is less than 50 kSLOC (thousands lines of source codes). Above this threshold, he might have to go back to his design documentation or dive back into the code.



Rule 3: 15 concepts can be ingested per day. Newell derives this quantity by measuring the learning process of a chess player. On average, a chess player can learn 15 "positions" per day. This is somehow related to the 50 000 limits because it takes on average 10 years of training to become a chess master (15 concepts * 360 days * 10 years is 54 000 concepts).
Rule 4: Individuals search "depth first". This is explained later.

3 System Engineering process revisited

This section revisits some process of the INCOSE handbook (ref 5). For each process, one or more Newell's rules are suggested.

3.1 CONOPS and FBS

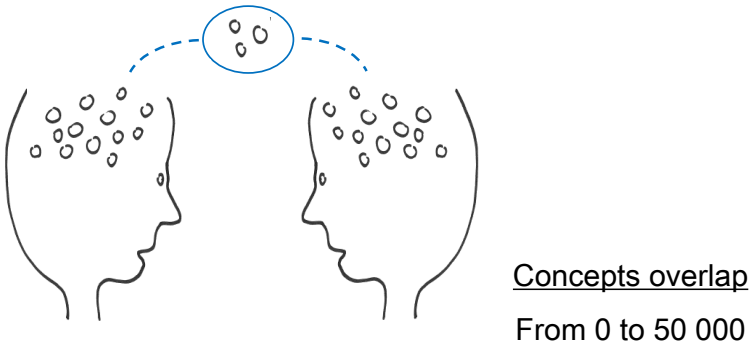
When System Engineers draft CONOPS, it is, through stories, to describe the system bit by bit. A rule of thumb is that a good CONOPS should be as interesting to read as a novel. With cognitive sciences words, a CONOPS should be an ergonomic device for pouring concepts into the mind of the reader.

From a completed CONOPS, System Engineers inventory all the functions elicited in the CONOPS and assemble them in an abstract hierarchical tree named the functional breakdown structure (FBS). In order to do that, System Engineers create abstract nodes that assemble functions and abstract nodes that assemble other abstract nodes. The top abstract nodes are the main functions of the system.

This abstract hierarchy makes it easier to understand the system in a top down view. The cognitive science justification is that our brain cannot manipulate more than 7 ideas simultaneously. Vertical hierarchies are fit for our limited intellectual span.

3.2 Requirement elicitation

When eliciting requirements for a system, two individuals coming from different disciplines try to contract on shared abstractions. One individual is the stakeholders agent. He represents the discipline that operates the system. The other individual is the designer. He represents the discipline that develops the system.



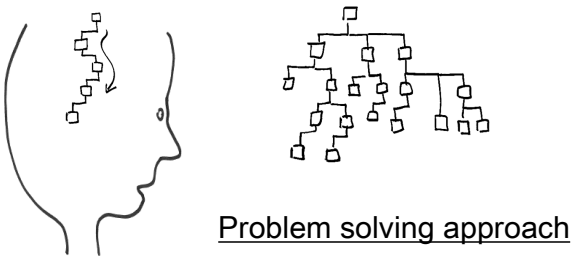
What effort shall we expect? Let us say we open the brain of both individuals, map both sets of 50 000 concepts and measure the overlap between both maps. This overlap may be large or thin. If it is large, a small effort is needed by the two individuals to elicit requirements (a lot will be implicit). If it is thin, misunderstanding might happen. The thinner, the harder both individuals have to explicit common concepts to create requirements.

3.3 Architecture process

In some cases, finding the best architecture is done with the help of mathematical tools such as operational research. Here, we are interested in cases when the architecture decision is driven by human cognition alone.

Since architecture is the most creative and diverse part of the System Engineering processes, it is preposterous to propose generalities. Yet, the article tries to relate one rule of thumb for system architects (the “no fly to solution” rule) to one rule elicited by Newell (the “depth first” rule).

According to Newell, when a brain is looking for a solution, it balances between two perpendicular directions, the vertical “depth first” direction and the lateral direction.



"Depth first", which is the most natural tendency, derives of Newell's observation of chess players. For every move, a chess player anticipates was will be the opponent response. From this hypothetical answer, he devises his own answering move. This problem is usually formalized as a tree exploration. Some chess master can explore up to 10 moves down the tree.

Newell notices that good players tend to explore the tree in "depth first" rather than by layers. A chess master explores many branches of the tree in sequence. He then creates a global view of the tree to choose the branch of the tree with the most interesting outcome. In other words, he explores tactically before elaborating a strategic picture.

In System Engineering, selecting the first available architecture is not recommended (the "no fly to solution" rule). INCOSE commends choosing choice criteria, then construe alternatives and then evaluate all alternatives against all criteria.

Cognitive sciences might suggest a different perspective. It is necessary to go deep to have insights into the architecture strategy. In other words, some selection criteria will not be present at the start, they will surface along the design exploration (we see later how recent trends in technical management are more aware of this fact).

3.4 Interface design

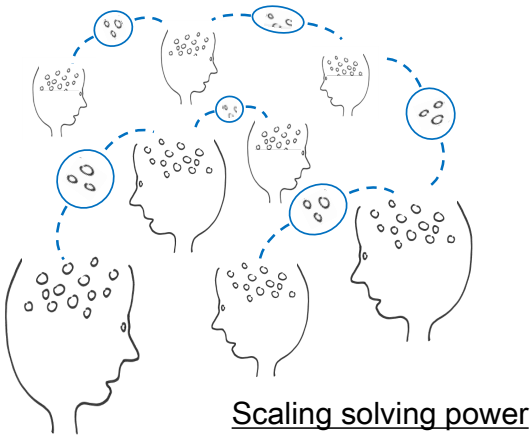
Architecture design decomposes a system into sub-systems with interfaces. When engineers design interfaces, they mitigate the integration risk.

Let us take two different specialty engineers who have to design an interface. Both need to share some of their 50 000 concepts. With requirement elicitation, we have already seen this situation where individuals need to share concepts. Instead of requirement elicitation, interface design is not an exercise in abstraction. Indeed, an interface has to be as detailed as possible to mitigate the integration risk between components.

However, in software engineering, a "good" interface design is an interface that aims at making the system across the interface the most abstract possible. A good interface limits the complexity for the interface users (there brain have to consume less concepts). For instance, the High Level Architecture (HLA) standard used in distributed simulation exhibits a "double abstraction barrier" to forbid participants to known too much about the other participants (ref 2).

3.5 Weaving abstract hierarchies together

Thanks to requirement elicitation, architecture and interface design, System Engineers can create Product Breakdown Structures (PBS). A PBS enables different teams to work in parallel on different part of the system. A PBS scales the engineering power with the number and diversity of contributing talents.



When System Engineers create PBS, it is to reflect the integration strategy. A PBS is one amongst many ways of putting system elements into abstract hierarchies, for instances:

- An operational expert categorizes people. He ends up describing an organizational breakdown structure (OBS).
- A roll-out/ILS team is interested in how to deploy and maintain the system. It devises a logistical breakdown structure (LBS).
- A Manager is interested in how the work will be done. He formalizes work breakdown structures (WBS).
- A security expert devises red area, black area, DMZ, trusted areas. He construes security breakdown structures (SBS).
- A financier relates work and procurement expenditures to his cost breakdown structure (CBS).

System Engineers trade is to weave those abstract representations of the system. A System Engineering Management Plan (SEMP) shall describe ways for teams to reach consistent breakdown structures with fluidity. For instance, at Airbus Defense & Space, our Model Based System Engineering (MBSE) tools are tailored to deal with overlapping hierarchies of data, including scheduling data.

4 Alternative approaches to technical management

In architecture & decision processes, INCOSE states that you should first gather all necessary information and then make the design/decision (see above).

Some other architecture/decision processes that have received attentions in recent years are a bit different, and perhaps more aware of our cognitive constraints. Toyota's Set Based Concurrent Engineering (SBCE, ref 3) and Lean Startup (ref 6, 7) are such exercises in architecture/decision process. Both surmise that some pieces of information necessary for decision are not present upfront, they come along the development process.

4.1 Set Based Concurrent Engineering

Toyota's SBCE decision process entertains more than one engineering solution down the development cycle. As stated by Sobek:

> In developing a vehicle's styling, Toyota makes more […] models than most competitors do. […] Simultaneous with the development of the two to three full-scale models, Toyota engineers develop structural plans for multiple styling design ideas and analyze them for manufacturability.

Sure, SBCE is more work upfront but less rework is to be expected downward. Indeed, SBCE is more robust to late problems discovery because more solutions are kept alive. Sobek contrasts SBCE with point based concurrent engineering where one solution is selected early and finding problems late leads to rework up the value chain. For instance at Airbus, at Preliminary Design Review (PDR), the usage is to have only one solution to be reviewed.

4.2 Lean startup

Another example is the "lean startup" agile approach. In SE classical view, people either plan or execute the plan. If something goes wrong, it is either the plan or the execution that was wrong. Lean startup suggests that when a system interacts with an environment that is complex and not controlled, the plan/execute model maybe not optimal.

When designing social web applications (the focus of Ries's work), the system interacts with thousands of individuals/clients. Then a simple *configuration management* issue kicks in. It is impossible to inventory and control the 50 000 concepts of the thousands of individuals that will use the system.

To fix the "brain configuration control" issue, Ries proposes to perform more but shorter "plan/execute" cycles. In a "lean startup", the business concept under study is declined in assumptions (let us say 20 assumptions for a typical concept). At the beginning of each cycle, a set of those assumptions is selected (let us say 5 assumptions). The objective of the cycle is to develop parts of the concept that enable the team to measure if those assumptions are true. If, at the end of the cycle, some as-

sumptions are wrong, the lean startup must be prepared to "pivot" and elicit a modified concept with new assumptions (that again need to be rigorously tested).

According to Ries, a Lean Startup measures progress not in units of work achieved but in units of verified learning.

5 Practical conclusions

5.1 Design of requirement documents

As System Engineers, we are interested in SMART requirements.

However, cognitive science hints that there is also a challenge elsewhere, the design of requirement documents in complex systems.

First, a requirement is a shared concept between two brains. A requirement is like a prosthesis between the brain of the stakeholder agent and the brain of the designer. Like any prosthesis, a requirement needs to be "adapted" to its users.

Second, a requirement is not alone. It is formalized in a document containing a set of requirements. Ideally, this document shall be approved by two persons only: the stakeholder agent and the designer. Therefore, a requirement document shall fit both the brain of the stakeholder agent and the brain of the designer.

In some System of Systems design, such as Air Traffic Management, stakeholders (operational experts) and designers (system architects "at system level") can share some requirements but it is difficult to group requirements into documents that span more than one system. Indeed, if a system is complex, a system designer can only elicit requirements with a span of one or two systems at best (cf. the "50 000 concepts" rule).

How to elicit requirements that have a span across all the systems into a single document ?

The solution to this cognitive constraint is to design requirements in two layers: a "system of systems" top layer and a "system" layer. The "system of systems" requirements are more abstract than "system" requirements. The "system of systems" requirements cannot be used directly for the design of system but they can fit into one document.

The design challenge is to find system engineers that can work sandwiched between both layers. Seen from operational experts, they are designers who can elicit requirements in a document with a span of all systems. Seen from the system architects "at system level", they are stakeholders who can elicit requirements, in the "system" layer, for each systems. The design challenge is a logistic challenge: to find people with a brain that have a good balance of 50 000 concepts, for instance 25 000 concepts shared with operational experts and 25 000 concepts shared with system designer.

5.2 Evaluating SE needed overhead

Another problem is to assess the need for System Engineering artifacts. Do we need to formalize an interface? Is a simulation needed to demonstrate the functional behavior of the system?

Let us take simulations. They can be expensive but they provide visual cues, the path with the highest throughput to the concept store in our brain. The case for simulation shall sometimes be a cognitive case. Are the stakeholder already experts in the operational view (weak case) or are they novice (strong case).

5.3 Teaching the INCOSE handbook

To pass an INCOSE certification, one rote learns 25 different processes. The global coherence is difficult to grasp. Clearly, outputs of some process are inputs to other process. Yet, it can be rather difficult to derive a general underlying argument to justify all processes.

The INCOSE backdrop philosophy is that it is good to put 10 to 15 % SE overhead to have a complex project succeed. This philosophy is pragmatic, it is based on regular observation of patterns in past complex projects but it is not a generative argument. This article suggests that Cognitive Science could be mentioned in SE handbooks introduction. It could answers “why do we need SE?” before going to the “what is SE?”.

An interdisciplinary point of view toward system engineering can be insightful. In 1984, Perrow, a sociologist, published his “normal accident theory” (ref 10) that challenges the traditional risk approach to complex system engineering. The book second chapter is titled: “Nuclear Power as a High-Risk system: Why we have not had more Three Miles Islands – but will soon”. It is, in retrospect, an insightful point of view.

6 Conclusion

If cognition is the mother of all constraints in System Engineering, it is time to measure cognition constraints consistently. In many occurrences, System Engineers rely on their experience and their “soft skills” to assess people constraints.

Between cognitive science and system engineering, an interdisciplinary academic effort could provide us with bespoke tools to assist system engineers in their daily work. For instance, we could evaluate more consistently the cognitive overlap between two individuals.

Academics sometimes testify that teaching SE to students can be challenging. As stated by Simon himself, the subject is abstract and a bit confusing (“cooky booky”). Simon compares here SE to other engineering discipline based on hard sciences (ref 8). If cognitive scientists could provide system engineers with quantitative tools, SE could become a more “normal” discipline.

7 Aknowledgment

The author acknowledges helpful and good humored discussions with Dominique Ernadote, Jean-Victor Noël-Betemps, Bruno Carron and Jean-Luc Wippler.

8 Bibliography

1. Newell, A. (1994). *Unified theories of cognition*. Harvard University Press.
2. Dahman, J., Ponikvar, D. R., & Lutz, R. (1996, March). HLA federation development and execution process. In *15th Workshop on Standards for the Interoperability of Distributed Simulations* (pp. 327-335).
3. Sobek, D. K., Ward, A. C., & Liker, J. K. (1999). Toyota's principles of set-based concurrent engineering. *Sloan management review*, *40*(2), 67-84.
4. Gautreau & al. (2013). Lessons learned from NMSG-085 CIG Land Operation demonstration. *Spring Simulation Interoperability Workshop,* paper 13S-SIW-031.
5. INCOSE System Engineering Handbook V3.2: A Guide for System Life Cycle Processes and Activities.
6. Ries, E. (2011). The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses. Random House LLC.
7. Blank, S. (2013). Why the lean start-up changes everything. *Harvard Business Review*, *91*(5), 63-72.
8. Simon, H. A. (1996). *The sciences of the artificial*. MIT press.
9. Newell, A., & Simon, H. A. (1972). *Human problem solving* (Vol. 104, No. 9). Englewood Cliffs, NJ: Prentice-Hall.
10. Perrow, C. (2011). Normal Accidents: Living with High Risk Technologies (Updated). Princeton University Press.

Capability-Based System-of-Systems Approach in Support of Complex Naval Ship Design:

Jacques P. Olivier[1], Santiago Balestrini-Robinson[2]
[1] Royal Canadian Navy
National Defence Headquarters, Ottawa, ON, Canada
jacques.olivier@forces.gc.ca

[2] Georgia Tech Research Institute
Georgia Institute of Technology, Atlanta, GA, USA
sanbales@gatech.edu

Abstract. Naval ship design can be understood to be a networked System-of-Systems (SoS) multidisciplinary process whereby a decision on one aspect of the design may have simultaneous, multiple order effects on other aspects of the design. Modern naval ship design should therefore consider the systems of interest as components subsumed by a holistic environment encompassing assets and capabilities inorganic to a naval platform. This position paper propose a starting point approach intended to provide a more defined means of establishing and improving the ship design process as part of a multi-layered maritime domain warfare enterprise. Fundamental is the tenet that capability levels transcend several hierarchical echelons and exist across many functional domains. The proposed methodology provides a structured and cohesive approach for identifying and assessing ship capability portfolio with traceable and better known impacts on mission effectiveness, affordability and risk, in the early stages of ship design within the scope of a naval system-of-systems.

1 Introduction

The ship design of major surface combatants capable of effectively responding to all possible missions within the spectrum of modern conflicts and military operations other than war is increasingly difficult due to the complex nature of the rapidly evolving and unpredictable global threat environment. Traditional naval ship design methodologies have evolved from the sequential nature of the design spiral to more advanced computational methods enabling the simultaneous manipulation of several degrees of freedom to better understand the interdependencies between factors such as cost fluctuations, design parameters, technology selections and mission success [1].

The issue persists nevertheless that although we may have a multidisciplinary team applying domain knowledge and experience onto systems engineering analysis, the optimization of the design process may remain restrained by designing ships within intrinsic ship capabilities as opposed to designing ships subsumed by a holistic environment encompassing assets and capabilities inorganic to a naval platform.

2 Motivation

The proposed method is intended to provide a more defined means of establishing and improving the ship design process as part of a multi-layered maritime domain warfare enterprise. To achieve this, the design approach is dependent upon high levels of confidence in the fidelity of the analyses, and is based on shared understanding and a common language.

Alike best practice in portfolio, programme and project management [2], using such an approach should deliver a range of benefits which will be revisited throughout the paper, these include:

- Identifying capability strengths and interests to be maintained, developed and exploited.
- Identifying capability deficiencies (shortcomings or surpluses) to be remedied or accepted.
- Providing a more structured and cohesive approach to identifying and assessing ship capability portfolio.
- Creating a common language and conceptual framework for the way to manage and improve capability-based planning within a ship design process.
- Educating stakeholders on the fundamental elements of capability-based ship design and how they relate to their roles and responsibilities.
- Involving more relevant stakeholders at all levels in the capability-based ship design process.
- Ranking ship variants based on operational effectiveness, capability and affordability trade-offs across a spectrum of missions' priorities.

- Facilitating comparisons, identifying and allowing the sharing of best practice across major ship acquisition projects within an organisation or a community of practice.
- Assessing and presenting the findings from a variety of reviews in a format that is easy to understand.

The aspiration is to show how these benefits can be realised through a combination of techniques including the adroit use of model-based systems engineering (MBSE): the formalized application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases [3].

Recognizing that MBSE has as its foundation the use of models, the approach is limited to construct an abstraction of selected aspects of the behaviour, operation, or other characteristics of a real-world SoS [4]. The purpose therefore is not to eliminate all uncertainties and cover all options related to ship conceptual design but to circumscribe them so to distil a deeper appreciation of the critical factors.

3 Naval Surface Combatants as System-of-Systems

3.1 SoS in Defence

Applications of systems engineering (SE) and SoS principles abound in Defence. Indeed, a growing proportion of the acquisition, sustainment, and management of materiel and non-materiel of military capabilities is sought through a SoS approach [5]. Moreover, the adoption of enterprise architectural framework in Defence by several nations is a definite step towards providing a more rigorous approach to life-cycle management including governance, design, building, analysing, and change management [6]. For instance, the UK Ministry of Defence Architecture Framework (MODAF) offers the following benefits within the acquisition processes [7]:

- Improved clarity on the context within which a new capability will operate.
- Clearer and more comprehensive requirements documents.
- Improved ability to resolve interoperability issues between systems.
- Better understanding of the mapping of system functions to operational needs and hence the ability to conduct improved trade-offs.

The proposed approach aims to utilize an architectural framework similar to MODAF to embody the SoS elements, unify their capabilities at the appropriate hierarchical levels, and define their interdependencies to provide a common picture of the SoS measure of effectiveness (MoE).

3.2 SoS in the Navy

Basic sets of architecting principles were proposed by Maier as discriminating factors to assist in the design of SoS [8], which later generated five characteristics that define SoS more appropriately [9]. This useful taxonomy may be used to draw the

SoS boundaries for a naval platform, namely: operational independence of the individual systems, managerial independence of the systems, geographic distribution, emergent behaviour and evolutionary development.

Recognizing that a single platform is a contributing element of a naval SoS, it follows that we should attempt to define the measures of effectiveness (MoE) of that SoS. In naval terms, models of hierarchical complexity could be translated into naval ranks and typology such as those described in Fig. 1 [10]. The legend could be used to characterise the MoE of a naval SoS hierarchically from a naval force capable of independently carrying out all the military roles on a global scale to that which has minimal ships' capabilities and is intended to only perform the most limited of constabulary functions. Mapping against the typology levels could facilitate the ranking of ship variants based on potential operational effectiveness and capabilities trade-offs across a spectrum of missions' priorities.

Rank	Typology	Naval SoS Description
1	**Complete Major Global Force Projection**	Capable of carrying out all the military roles of naval forces on a global scale. It possesses the full range of carrier and amphibious capabilities, sea control forces, and nuclear attack and ballistic missile submarines, and all in sufficient numbers to undertake major operations independently.
2	**Partial Global Force Projection**	Possesses most if not all of the force projection capabilities of a "complete" global navy, but only in sufficient numbers to undertake one major "out of area" operation.
3	**Medium Global Force Projection**	May not possess the full range of capabilities, but have a credible capacity in certain of them and consistently demonstrate a determination to exercise them at some distance from home waters, in cooperation with other Force Projection Navies.
4	**Medium Regional Force Projection**	Possesses the ability to project force into the adjoining ocean basin. While may have the capacity to exercise these further afield, for whatever reason, do not do so on a regular basis.
5	**Adjacent Force Projection**	Possesses some ability to project force well offshore, but not capable of carrying out high-level naval operations over oceanic distances.
6	**Offshore Territorial Defence**	Possesses relatively high levels of capability in defensive (and constabulary) operations up to about 200 miles from shores, having the sustainability offered by frigate or large corvette vessels and (or) a capable submarine force.
7	**Inshore Territorial Defence**	Primarily inshore territorial defence capabilities, capable of coastal combat rather than constabulary duties alone. This implies a force comprising missile-armed fast-attack craft, short-range aviation and a limited submarine force.
8	**Constabulary Defence**	Not intended to fight, but to act purely in a constabulary role.

Fig. 1. Naval System-of-Systems Levels.

4 Capability-Based Framework

4.1 Capability Definitions

Military concepts generally use a lexicon of frequently interchangeable terms with sometimes only subtle differences in meaning and often dependent entirely upon context. For instance, words such as mission, role, function, task, activity, and capability may have both a descriptive sense ("what") and a process sense ("how"). The descriptive sense defines the purpose or basic functions of an organisation and identifies the precise nature of an operation to be conducted in pursuit of an assigned mission or objective. The operational sense denotes the precise activities to be undertaken or achieved which in combination contribute to mission success [10].

It is recognized nevertheless that there are those essential capabilities which are common to any naval force at any time, as required to exercise any of the missions, roles, functions or tasks that might be assigned to it. The degree to which these core competencies are required and met is predicated upon the needs of the local and temporal situation. Ergo, they will be considered as capability priorities summarised by the basic naval concepts of float, move and fight for the purpose of this study.

Of note, the United States Department of Defense (US DoD) defines a capability as the ability to achieve a desired effect under specified standards and conditions through combinations of "ways" and "means" to perform a set of tasks [11]. This definition joins the previous definitions in that the "ways" are the strategic and operational methods describing "how" to conduct military operations to accomplish the specific military objectives, the "ends", while the "means" describe "what" resources are adequate to achieve these objectives within an acceptable level of risk.

Lastly, the level of operational capability and the potential response time constitute the basis for the concept of readiness which is a measure of the ability to undertake an approved task, at a given time. Four readiness levels are considered in this study [5]:

- Extended Readiness (ER): Not operational.
- Restricted Readiness (RR): Transitioning between readiness levels or subject to deficiencies in personnel, materiel and training severely limiting employment.
- Standard Readiness (SR): Capable of conducting core naval continental and expeditionary missions that do not entail the possibility of high intensity, full spectrum combat.
- High Readiness (HR): Capable of conducting the full-spectrum of combat operations.

As will be seen in the next section, these definitions may be used to create a common language and conceptual framework that may facilitate identifying capability strengths and interests to be maintained, developed and exploited; but also identifying capability deficiencies (shortcomings or surpluses) to be remedied or accepted.

4.2 Cross-Functional Capability Framework

This position paper espouses the tenet that capability levels transcend several hierarchical echelons and exist across many functional domains. For instance, from a marine platform systems viewpoint, a hierarchy of equipment-based capabilities prescribe the minimum materiel standard necessary to support the intent of materiel safety [12]. That baseline level identifies the equipment that must be available for ships to proceed and remain at sea, i.e., float and move capabilities in higher than restricted readiness. Other equipment may now be selected to enhance the platform systems capability levels or elevate the combat systems capabilities enabling fighting at the standard or high readiness levels.

Fig. 2 and Fig. 3 illustrate examples of capability-based frameworks showing how materiel availability at the equipment level could be mapped to operational effectiveness using the definitions offered for temporal capability priorities, platform and combat systems capabilities, and operational capability readiness levels.

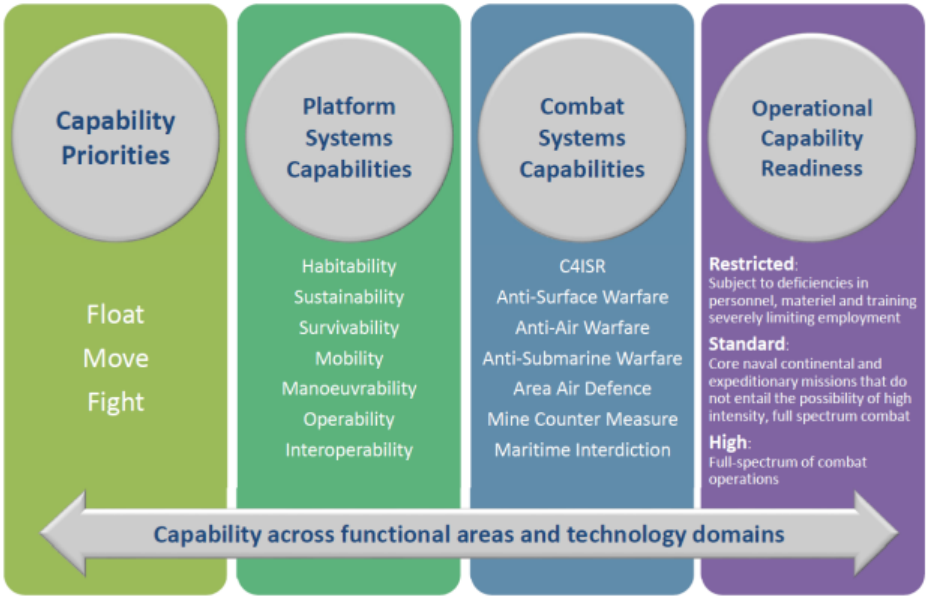


Fig. 2. Cross-Functional Capability Framework – Example 1.

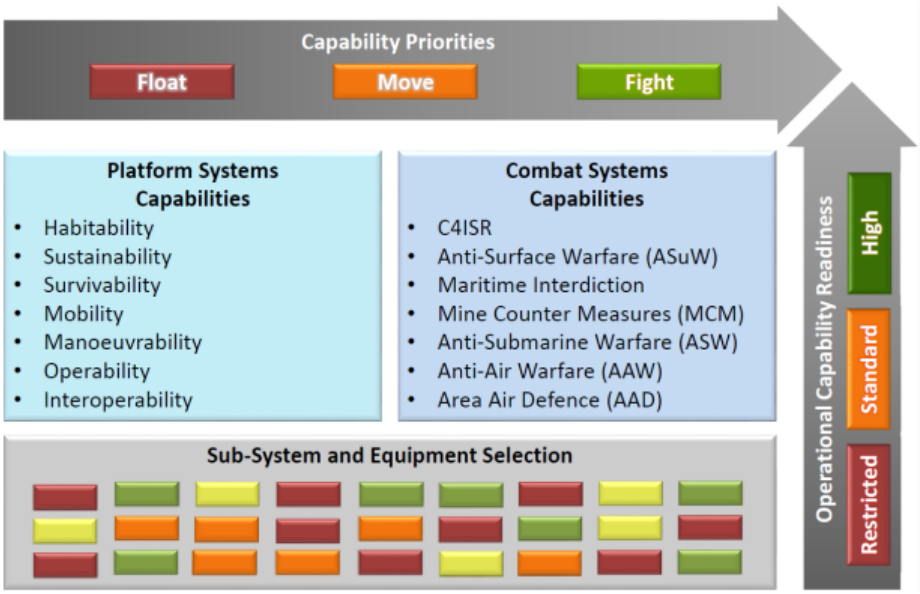


Fig. 3. Cross-Functional Capability Framework – Example 2.

4.3 Capability-Based Planning

Concepts of capability-based planning (CBP) in enterprise architecture can be invoked to explain that capabilities can be horizontal, going against the grain of business processes (platform and combat capabilities), or be vertical, being handled in the context of the business organizational structure (task group, flotilla or squadron) [13]. Applied to the military context, CBP evolved from threat-based planning, and is envisaged as the framework that will permit the military forces to optimize their capacity to respond to the range of plausible missions in which they may be called upon to serve.

CBP is a systematic approach for identifying the levels of capabilities needed to meet government priorities. Using scenarios, CBP explicitly connects capability goals to strategic requirement to develop force options more responsive to uncertainties, economic constraints and risk [14]. CBP is thus not estranged to the Defence realm and its principles were used as a pillar to the proposed ship design methodology.

5 Methodology

5.1 Hierarchical Capability Decomposition

Inspired from hierarchical functional decomposition (HFD) principles, the proposed approach suggests to decompose, prioritize and recompose capability requirements through the strategic, operational, tactical and technical levels of abstractions enabling both the descending “top-down” approach from political aspirations and the ascending “bottom-up” approach from equipment-level capabilities. The hier-

archy can span any set of functional levels, but it should always include as its lowest level a tangible set of requirements that can be mapped to physical systems and performance constraints. The process generates upward, lateral and downward connections to produce a collectively created and shared picture of the SoS being designed.

These ship-level platform and combat systems capabilities, which correlate to system-level key users requirements, could be mapped to tactical-level capabilities usually pertaining to effectively conduct a combination of naval functions under prescribed conditions, with other SoS elements. The overall achievement of naval functions would subsequently propagate up the hierarchy to analyze the effect that a given set of ship systems capabilities have on higher level operational and strategic capabilities.

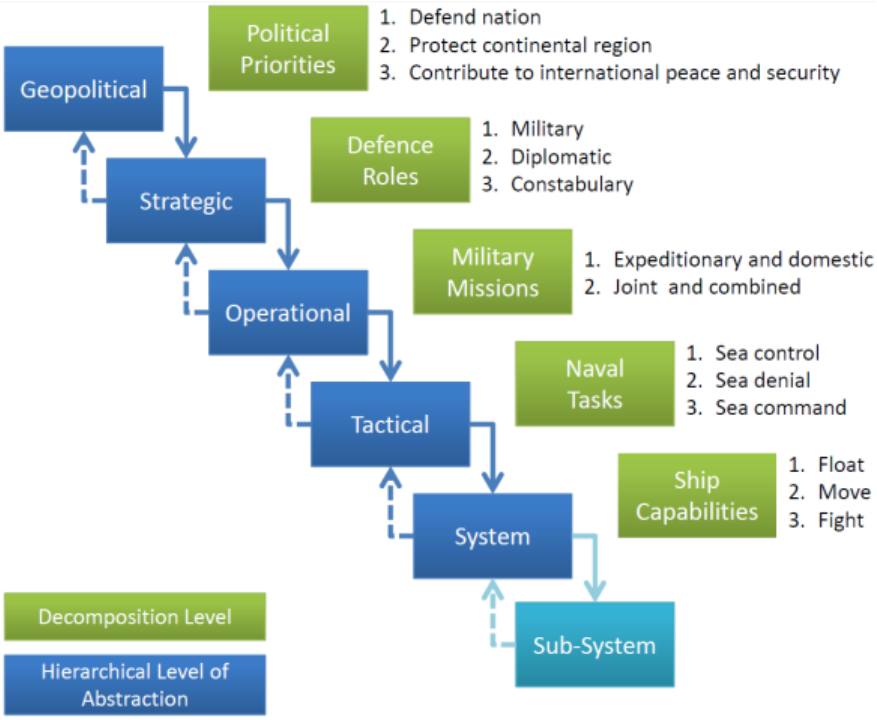


Fig. 4. Approach to Hierarchical Capability Decomposition.

As shown in Fig. 4, the process involves eliciting capabilities by mapping and prioritizing strategic-level defence roles with operational-level domestic and expeditionary missions which are in turn linked to tactical-level naval functions. These naval functions are the bridge to ship-level capabilities where the SoS is decomposed into its elements by systems, sub-systems and equipment.

5.2 Interactive and Dynamic Capability-Based Trade Studies

This approach creates a dynamic SoS architecture decomposing and linking high-level organizational goals to key performance parameters. By integrating all design analyses, including cost models, into a single environment, probabilistic methods and surrogate models can be used to facilitate parametric trade studies and capture the propagated uncertainties impacts.

As summarized in Fig. 5, the interactive and dynamic trade-off studies will results in design variants at the ship-level capabilities which better define the performance of the ship independently of mission scenarios, or as an element of a SoS, in the early stages of ship design. It follows then that when taken as an element of a SoS, much consideration is applied to create a solution with a higher MoE. The equipment and systems-level study will generate better key user requirements selected on merit because they are critical to the achievement of operational needs and the appeasement of political pressures.

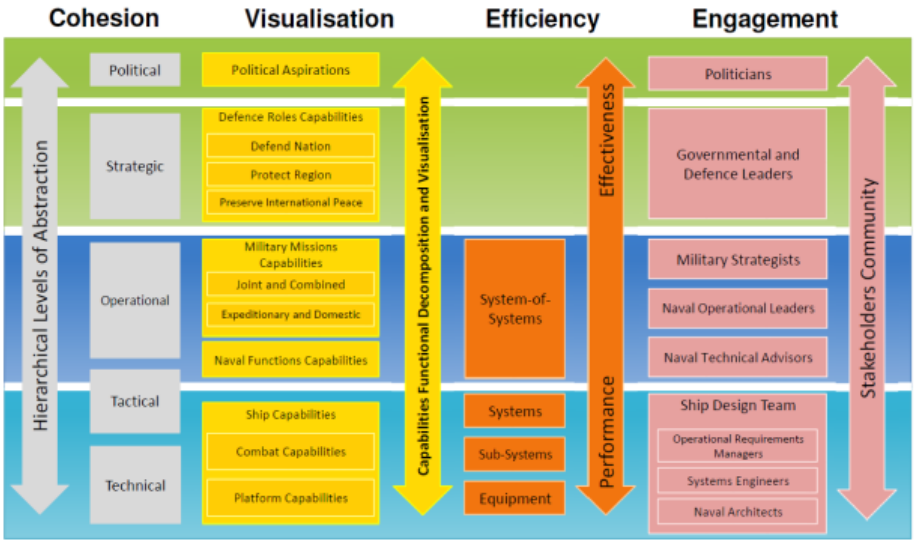


Fig. 5. Capability-Based SoS Approach for Ship Design.

The intent of the capability analysis is to capture the knowledge and experience of the subject matter experts (SMEs), so as to allow a decision maker to assess a large number of potential ship capability combinations without the need to query the SMEs each time.

One of the objectives is to unify the stakeholders' community such that a naval architect can readily understand the impact of a design configuration or equipment selection on the effectiveness to achieve a specific mission at the SoS level. Conversely, a strategist may better understand the technological implications of privileg-

ing a given political defence priority. By involving more relevant stakeholders at all levels, greater awareness and education may be reached on the fundamental elements of capability-based ship design and how these canons relate to the stakeholders' roles and responsibilities.

5.3 Visualization

Communicating the potentially complex fused common operating picture encompassing the interdependencies between domains and disciplines to the stakeholder community is essential to sharing a collective understanding of the issues. The use of dashboards is an obvious first choice as they are visual displays that can often communicate with greater efficiency (can be more intuitive) and have richer meaning than text alone. Moreover, as exhibited in Fig. 6, a well-designed and customized dashboard would summarize the information most needed to achieve specific objectives in a single screen using clear and concise displays mechanisms that are easy to comprehend [15].

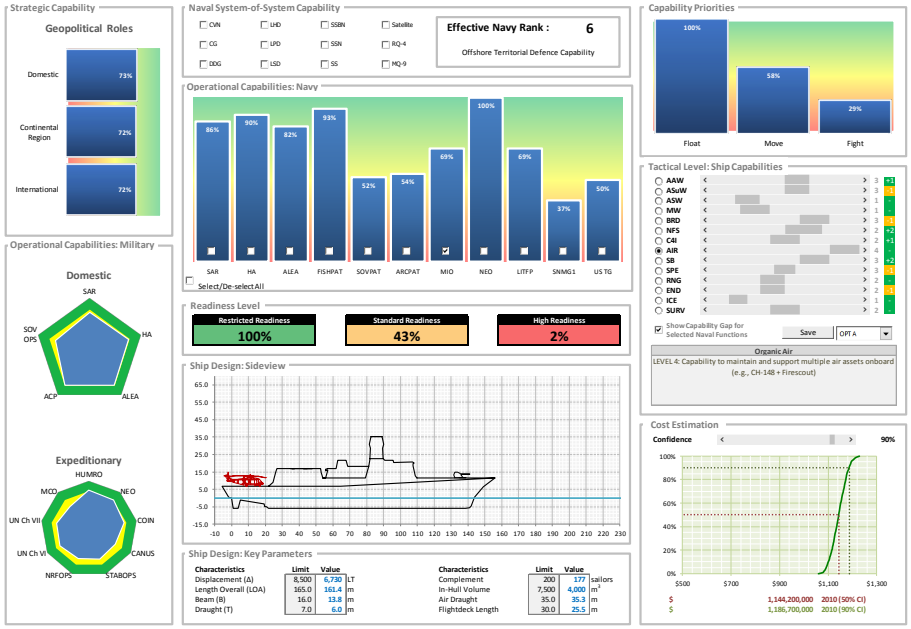


Fig. 6. Ship Design Synthesis Dashboard

These visualization methods may assist assessing and presenting the findings from a variety of reviews in a format that is easy to understand. Ultimately, the visualization of these outcomes provides the catalyst for decision-makers to more confidently consider options they would otherwise ignore and move forward based on well-founded assumptions.

It is acknowledged that verification and validation of the characteristics and behaviours of the SoS comply with the design intent is usually performed while the systems are being integrated and upon completion of sea trials and ship acceptance. But as earlier stated, correctly applying MBSE methods within an architectural framework may improve the ability to resolve interoperability issues between systems, and improve clarity on the context within which capabilities will operate.

6 Conclusion

This position paper proposed an initial approach intended to provide a more defined means of establishing and improving the ship design process as part of a multi-layered maritime domain warfare enterprise. The proposed methodology provides a structured and cohesive initial way forward for identifying and assessing ship capability portfolio with traceable and better known impacts on mission effectiveness, affordability and risk, in the early stages of ship design. The epistemic nature of the proposed process allows the collective generation and evaluation of scenarios which challenges prevailing mind-sets and presumed correlations between uncertainties, while reducing subjective interpretations. Again, the purpose is not to eliminate all uncertainties and cover all options related to ship conceptual design but to circumscribe them so as to instil a deeper appreciation of the critical factors.

7 Disclaimer

This paper is an unclassified position paper containing public domain facts and opinions, which the authors alone considered appropriate and correct for the subject. It does not necessarily reflect the policy or the opinion of any agency, including the Government of Canada, the Canadian Department of National Defence, or the Georgia Institute of Technology.

References

1. J.P. Olivier, S. Balestrini-Robinson, and S. Briceño, “Ship Cost-Capability Analysis using Probabilistic Cost Modeling and Hierarchical Functional Decomposition Methodologies,” 11th International Naval Engineering Conference and Exhibition (INEC), Edinburgh, UK: Institute of Marine Engineering, Science and Technology, May 2012.
2. Her Majesty’s Treasury, “Portfolio, Programme and Project Management Maturity Model (P3M3®) Introduction and Guide to P3M3®, London, UK: The Office of Government Commerce (OGC), 2010.
3. International Council on Systems Engineering (INCOSE), “INCOSE Systems Engineering Vision 2020 (INCOSE-TP-2004-02-02.03),” San Diego, CA, USA, September 2007.
4. Institute of Electrical and Electronics Engineers (IEEE), “IEEE Standard Glossary of Software Engineering Terminology (IEEE-Std 610.12-1990),” New York, NY, USA, 28 September 1990.
5. Canada, Department of National Defence, “CFCD 129 Readiness and Sustainment Policy,” Halifax, NS: Canadian Fleet Pacific Headquarters, October 2009.

6. Canada, Department of National Defence, “Department of National Defence and Canadian Forces Architecture Framework (DNDAF) Volume 1: Overview and Definitions, Version 1.8.1.,” Ottawa, ON: Directorate of Enterprise Architecture (DEA), 25 January 2013.
7. Her Britannic Majesty’s Government, “Ministry of Defence Architectureal Framework (MODAF) Acquisition Integrated Project Team (IPT) Community of Interest Deskbook Version 1.0,” London, UK: 31 August 2005.
8. M.W. Maier, “Architecting Principles for System of Systems,” 1999 John Wiley & Sons, Inc. Syst Eng 1: 267.284, Chantilly, VA, USA, 1998.
9. A.P. Sage, and C.D. Cuppan, “On the Systems Engineeing and Management of Systems of Systems and Federations of Systems,” Information, Knowledge, Systems Management 2(4): 325-345 (2001), Fairfax, VA, 2001.
10. Canada, Department of National Defence, “Leadmark: The Navy's Strategy for 2020,” Ottawa, ON: Directorate of Maritime Strategy, 2001.
11. Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering, “Systems Engineering Guide for Systems of Systems,” Version 1.0. Washington, DC: ODUSD(A&T)SSE, 2008.
12. Canada, Department of National Defence, “NAVORD 3000-0 Materiel Baseline Standard (MBS) – Surface Ships-Policy,” Ottawa, ON: Royal Canadian Navy, NMPRO 2/ACOS NEM, 23 April 2013.
13. The Open Group Architecture Forum, “TOGAF® Version 9.1 Enterprise Edition – Par III-32 Capability-Based Planning,” San Francisco, CA, USA: December 2011.
14. The Technical Cooperation Program (TTCP), Joint Systems and Analysis Group, “Guide to Capability-Based Planning,” TR-JSA-TP3-2-2004. Alexandria, VA: October 2004.
15. S. Few, “Information Dashboard Design: The Effective Visual Communication of Data,” 1st ed., Sebastopol: O’Reilly, 2006.

Probabilistic System Summaries for Behavior Architecting

Michael Borth

Embedded Systems Innovation by TNO,
PO Box 6235, 5600 HE Eindhoven, The Netherlands
Michael.Borth@tno.nl

Abstract. Smart system of systems adapt to their context, current situation, and configuration. To engineer such systems' behavior, we need to design and evaluate system-level control strategies and the intelligent management of key scenarios. We propose a model-based approach called *probabilistic system summaries* to explore related design choices, e.g., where to put the 'smarts' of a smart building. Our approach uses Bayesian inference to calculate effects of strategies and implementations, offering causal analysis of the costs and benefits of decision strategies in key scenarios. As the modeling is light-weight and suitable for various abstraction levels, probabilistic system summaries are appropriate for early but sound architecture decisions based on computational science. Next to its use within this analysis, the product of this engineering step, i.e., a Bayes net summarizing systems plus their environment, may form the core of decision making within the system of system..

1 Engineering of Smart Buildings

Smart systems of systems are set to recognize and adapt to their situation, to their operational context, and to their configuration, and to enact smart strategies to reach their goals in the best possible way. One example of such systems are smart buildings, where building automation is established to detect environmental circumstances, faults, utilization, or emergency situations, and then to act accordingly. They can, e.g., adapt building operations to energy saving demands while tuning services to the number of people present in a room and furthermore compensating for services that are not available due to failures. Such buildings are systems of systems: lighting, HVAC (heat, ventilation, air conditioning), security, etc. are individually developed and commissioned systems that operate independently, fulfil different and partially conflicting goals of various stakeholders, but benefit from cooperation. Especially an exchange of information is beneficial, e.g., on the presence of people while balancing comfort with energy savings. Complexity and size of building automation systems and high costs of commissioning make it desirable to realize building automation in a self-organizing, cooperative, and robust way.

These advanced functionalities require among other things I) the means to monitor the building and its environment, including the ability to detect faults and special events, II) system-level control strategies and scenario-management that allow the handling of and adaptation to foreseen circumstances, and III) solutions to handle unforeseen dynamics, e.g., fault adaptive behavior or an adaptation to requirement changes. In this article, we define our system-of-systems engineering method for the design, analysis, and architecture of behavior w.r.t. II), i.e., the model-based approach *probabilistic system summaries* that we use to investigate design alternatives and their trade-offs within the engineering of the 'smarts' of a smart system of systems.

1.1 Behavior Architecting within System of Systems Engineering

Following the argument that system of systems engineering (SoSE) focuses on choosing the right systems and their interactions to satisfy the requirements and goals of various stakeholders, we consider the architecting of behavior as a core task of SoSE. The focus on architecting, and not the mere design of functionality, acknowledges that the 'where and when' of a decision taken by a system matters - especially for systems of systems (SoS). Imagine, e.g., two alternate system control strategies in smart buildings: smart rooms that take local, if cooperative, decisions versus a smart building with centralized control. This architectural choice impacts among other things the need for data exchange and thus communication means (affecting costs and privacy), robustness (reliability of many cheap components or single point of failure), but also which data is available to adapt behavior and thus functionality itself.

An investigation of architectural alternatives that targets smart behavior requires methods and tools that enable a lightweight modeling of the solution candidates linked to computational science. It should allow the calculation of key performance indicators with regard to both the SoS' functional and non-functional aspects, e.g., the energy savings achieved by a decision strategy for efficiency or costs of operations. In this, we require a lightweight method, as a detailed process for each alternative incurs undue costs; computational science, i.e., modeling with quantitative analysis, is needed to discriminate decision strategies and architectures with regard to their expected business value, e.g., total costs-of-ownership.

1.2 Use Case: Where to Put the Smarts of a Building?

We support the introduction of the *probabilistic system summaries* method with a simplified case study, the investigation of the two architecture alternatives mentioned above, which addresses a central architectural question on decision making – where to locate it: locally, close to sensors and actuators, thus in many places with a limited range of impact but with direct communications, or in a central instance? Worded differently within the example, this is a choice between a smart building in which rooms without intelligence enact the building's strategy, or smart rooms that take local data to produce local decisions, while occasionally taking global information into account, which is provided by the building.

Most sensor data is local, about an individual area, e.g., a room, but global data always plays a role as well, e.g., outside temperature or available power. Actuators are often local, e.g., lights or air conditioning within a room, but shutting down the heating system is a global action. Strategies and goals might be individual and local, e.g., about the intended level of comfort in a room, or global, e.g., a reduction of energy consumption. The latter aspects, comfort and consumption, will serve as performance indicators of our use case. They represent conflicting goals of different stakeholders, i.e., building managers seek energy efficiency and thus minimal consumption (best achieved by turning everything off), while inhabitants seek comfort, possibly with very in-efficient ideas how to achieve that.

While this over-simplifies the architectural task and ignores performance indicators like costs or robustness of a realization, we believe that the reduction to two opposing indicators and two distinct architectural concepts for decision making and control works well for the demonstration of our design and analysis method. System architects working with our method will be able to define and add further performance indicators to suit their engineering tasks. In any way, we need to point out that the question which architectural solution to choose has no answer that holds for all possible buildings and possibly not even one that holds within one building for all situations. It must be investigated for each SoS – as any such decision impacts costs for operations and the infrastructure, and furthermore depends on the goals, situation, configuration, and many factors more, including non-functional aspects.

In our work, we also look into different AI techniques for smart self-organizing building automation, described in documents and papers listed at [1], with details on the aspects elaborated in this article in [2]. Such choices are not covered here. Instead, we assume that a technique is available and investigate its effects. Still, it is notable that certain architectures favor certain AI techniques to implement the needed smarts, e.g., local decisions with loose coupling work well with agent-based technologies.

2 Foundations: Bayesian Modeling

The decision between architectural alternatives warrants an investigation of each of them with regard to their costs, benefits, and effects. To avoid costly experimentation, we base such investigations on model-based computational science, with simulation, calculation, and probabilistic assessment as possible techniques. We opt for Bayesian networks [3] as probabilistic models. These graph-based representations of the joint probability distribution over all modeled variables offer causal probabilistic modeling [4], optimal to investigate cause-effect relationships, e.g., what impact a strategy has on energy consumption, given that probabilistic factors, like weather or the building usage, affect the outcome. As their calculus allows for the inference of probability distributions over variables given evidence or assumed circumstances, it becomes feasible to investigate the range and likelihood of possible outcomes. Furthermore, Bayes nets allow sensitivity analyses to determine the possible impact of factors [5], e.g., to investigate the dependency of a system's performance on the environment.

While the literature listed above provides extensive knowledge on Bayesian networks, the remaining article only assumes familiarity with the core concepts: Bayes nets are graphs with (random) variables as nodes. Directed edges between nodes show their relationships, i.e., probabilistic or causal dependencies, encoded as conditional probability distribution of a variable given all its parents. Causal relations are often functional: Given the cause, the effect is determined in a functional manner, with the conditional probabilities that encode the likelihood of a variable's states given other variables' states either at 0 or 1 – *impossible or always* true under the specified conditions. Probabilistic dependencies encode influence that is more random and flexible, e.g., a different likelihood for room occupancy given that the building is nearly full in comparison to situations where the building is mostly empty. Bayesian networks may either be modeled manually via knowledge engineering, or learned from data. Together with their real-time capabilities, this makes them suitable for many domains.

Several modeling techniques exist that enable the efficient use of Bayes nets for system modeling, e.g., by supporting re-use with object-orientation [6], and semi-automated construction of networks from knowledge bases [7] or system descriptions [8]. All this ensures a reduction of efforts as well as consistency: As we set out to compare strategies for system-level control for a given building that has a given environment, we use identical network building blocks, called *network fragments*, for the fixed points of our analysis, adding individually developed fragments for the variable parts. As all these fragments are constructed according to the same principles, they can be merged easily into a full Bayesian network.

3 Probabilistic System Summaries

In this section, we detail our probabilistic modeling that renders investigations of architectural alternatives for smart behavior. Essentially, the approach consists of a methodology to construct, in a comparable fashion, a Bayes net for each control strategy under investigation together with its context, i.e., a summary of the system and environment. Special nodes in the networks allow the computation of utility values of the control strategy, e.g., for energy consumption or comfort, given the assumptions encoded in the networks, since the Bayes nets represent the joint probability distribution over all modeled variables. The modeled strategies thus allow for experiments, e.g., to compare the utility values of various strategies given different assumptions.

3.1 System Summaries for Efficient Investigations

Our goal is to encode a global view on the system, in our domain a smart building, so that the impact (cause and effect) of strategies may be investigated. This global view requires insights in the distribution over possible values of key variables in the sense of a summary: There is, e.g., one variable for room occupancy and not one for each room. A distribution over this variable together with dependent variables is sufficient; for example, one needs to know that 70% of the rooms are occupied within a given scenario and that this level of occupancy results in a certain energy need.

Such a causal and probabilistic modeling that encodes a global view at the building allows for small models. This limits modeling efforts, especially compared to simulations that include individual rooms. However, even small networks hold a great number of parameters: With discrete states as possible values of the modeled variables, a node in the graph holds one parameter per state for each possible combination of the states of all parent variables. This leads to thousands of parameters in bigger networks. However, parameters may be computed automatically for functional dependencies, greatly reducing efforts. For non-functional dependencies, parameters are determined via a knowledge engineering process: estimates of the parameters stem from statistics, e.g., on room occupancy, or from an evaluation by experts. The object-oriented approach to model allows doing so in a constructive manner: a fragment encapsulates a piece of the domain, so that its modeling provides no challenge, as its relationships are easily understood. The subsequent construction of the Bayesian network from fragments follows a strict methodology, which may be used manually or even semi-automated, e.g., from a knowledge base (see references above and Chapter 7 of [9] for details of fundamental modeling techniques).

3.2 Use Case: Network Fragments for Smart Buildings

For an easily understood illustration of our work, we start with the following set of probabilistic variables (*in italics*) and the dependencies between them:

a) *Usage of the building*, a measure of utilization that impacts the likelihood of *room occupancy*.
b) *Condition of room*, a measure how outside factors set a room status, e.g., the temperature due to direct sunlight. It impacts, together with *room occupancy*, the *need of the room* with regard to services that consume energy, e.g., to cool the room down.
c) *Services provided* for a room, a measure of service-level provided for a room, which depends on *room occupancy* and the *need of the room*. (Furthermore the policy, not shown, which is not probabilistic, but user determined, as it describes a decision, e.g., to save energy. See below.)
d) The *consumption of the room* follows from the *services provided* to the room.
e) The energy *consumption of the room* determines the energy *consumption of the building*. The latter is a utility variable, i.e., a variable that shows a decision's utility value, in this case the aforementioned policy.
f) *Discomfort* within the building, another utility variable, which results from the discrepancy between the needs and services provided in occupied rooms.

Fig. 1 depicts the network fragments for these variables. It becomes visible that the fragments can be fused into a network that describes a building: conditions and occupancy determine need, services respond to need and consume energy, and a delta between need and services results in discomfort in occupied rooms. The probabilities are easy to determine: with statistic for variables without parents, with distributions for probabilistic relations, with causality for functional relations.

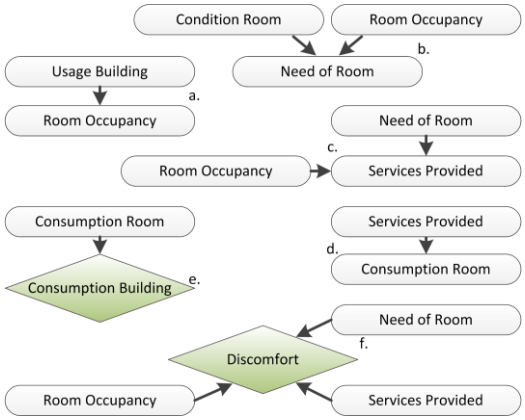


Fig. 1. Network fragments of simplified smart building use case

More elaborate models might mirror reality with less abstraction, e.g., by distinguishing between offices and meeting rooms instead of summarizing all rooms. This is, however, for the system architect to consider: a probabilistic system summary may be generated from known details as well as from more general insights. The network's causal nature ensures that different abstractions work together.

3.3 System-level Control Strategies

The decision process, i.e., the smarts of the building, is not explicit in the network fragments we introduced above. While it could be encoded within fragment *c*, which models the services provided to a room, this would hide the reasoning which is the goal of our investigation. Instead, we compose fragments that mirror the decision strategy and thus complement the other fragments that mirror the environment and processes. These fragments encode the causal effects of the strategy given the data available to the decision taker, incl. the current energy saving policy, summarizing both the decisions based on the available data and its effects on the system.

To analyze different strategies comparatively, we require a model for each strategy. The network fragments forming these models have two distinct aspects: the reasoning process, encoded in the network structure, and probabilities that define the strategy's parameters. The structure is engineered from the understanding which information is processed how within the system for decision takings. This might include complex steps, e.g., to account for missing data where the building automation has to use estimates. The strategy's parameters, on the other hand, are typically computed or determined with experiments, e.g., to find a policy set-point for services given a certain context so that a required energy saving is ensured. The modeling of the control strategies follows the same probabilistic summary techniques as the modeling of the environment and system interactions. It is, e.g., sufficient to know a distribution over alternatives for services set-points, disregarding where exactly a set-point is in effect.

3.4 Use case: Network Fragments for Control

In continuation of the simplified example, we show two distinct ways of building automation. Fig. 2 depicts the Bayesian network for smart rooms that take local decisions while considering global requests: Given local data on *room occupancy* and the *condition of the room*, the *need of the room* is determined. Given that and a *policy on energy savings*, the room sets the *services provided*. *Usage of the building* sets the distribution of the *room occupancy*. Utility nodes on *discomfort* and *consumption* follow functionally. This network holds close to 400 parameters. Fig. 3 shows a smart building that enacts a global strategy, and thus altogether a very different information flow and computational model. Here, information on the *needs of the rooms* and thus the *needs of the building* is collected, the outcome is compared to the constraints set by the *energy saving policy*, which leads to insights into *required savings*, that are then used to set an appropriate *strategy* that fulfils the savings while minimizing *discomfort*. This latter step exploits the information available through the collection of the room data: If, e.g., the calculation shows that the use of an aggressive saving strategy in empty rooms is sufficient to meet the requirements, discomfort in other rooms may be avoided altogether – a fact that an individual smart room could not have taken into account. The network holds over 1000 parameters. Fig. 4. shows a set of parameters that define set-points on how an energy saving policy for smart rooms changes the services provided to a room given its needs. It is key to understand that the complete models handle summaries in the form of distributions: if such conditions are given, this mixture of decisions will be taken, resulting altogether in these effects.

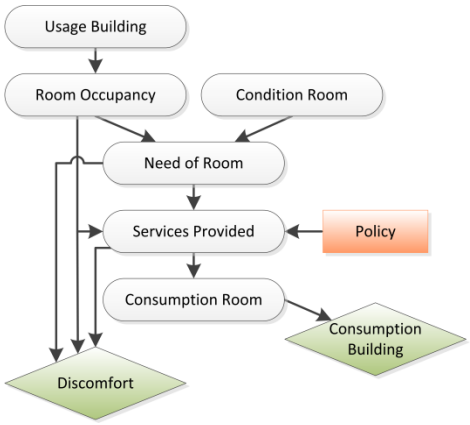


Fig. 2. Simplified Bayesian network for smart room building control

3.5 Network Formation

The final step in the modeling process is the formation of the network structure, i.e., the construction of the graph from the network fragments and other necessary nodes. This process, which may be partially automated [8], mirrors the information flows of the strategy and its realization, i.e., functional aspects as laid out above and other

architectural aspects. Fig. 2 and Fig. 3 show the out-come of this process. In these networks, we thus see a local decision process in smart rooms that deduct the services provided from locally available information while taking the global energy policy into account, while the smart building decides its strategy after global budget calculations.

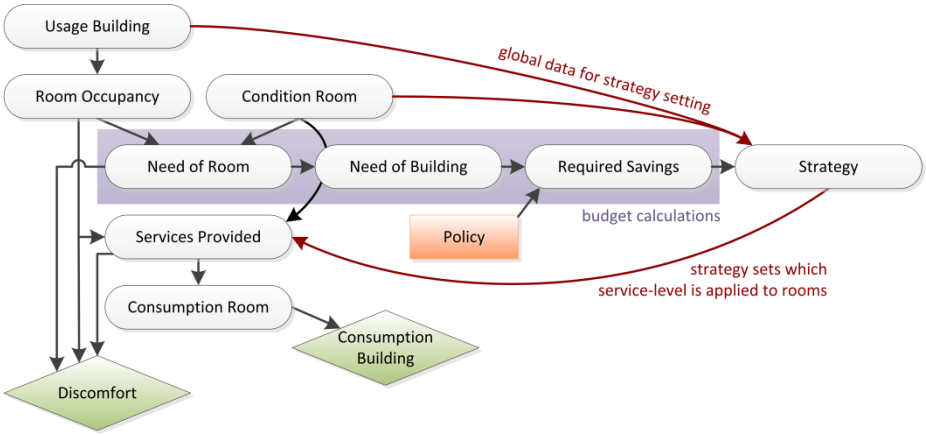


Fig. 3. Simplified Bayesian network for smart central decisions building control

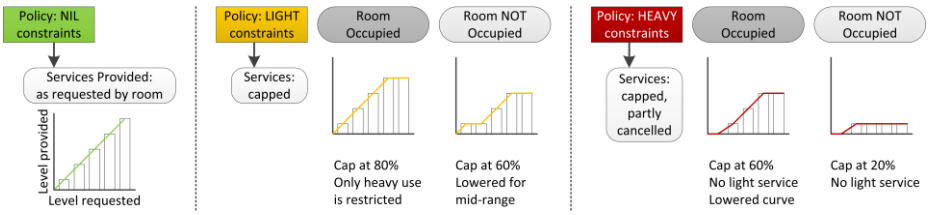


Fig. 4. Strategy for smart rooms: *services provided* given *occupancy*, need of room (*level requested*) for three energy saving policies (NIL, LIGHT, and HEAVY constraints)

If we encounter non-functional concerns, especially w.r.t. the realization architecture, we approach the modeling of their impact on the reasoning in the same way. Imagine, e.g., a masking of occupancy information for a building section due to privacy concerns, which results in incomplete information in the respective node. If the building's control is realized without mechanisms to compensate for this, we would soften the distributions of the room occupancy to allow for a wider range of values within our calculations. If the building's control has a setup to estimate these figures, we would include this flow of information, resulting e.g., in a room occupancy that follows from a combination of observations and an estimate model, which might be based on date and time, or car park observations. Fig. 5 summarizes workflow and considerations of the network generation process: The *Reasoning Network* encodes the information flows and functional aspects. If the realization architecture warrants an adaptation, the additional steps result in the final Bayesian network.

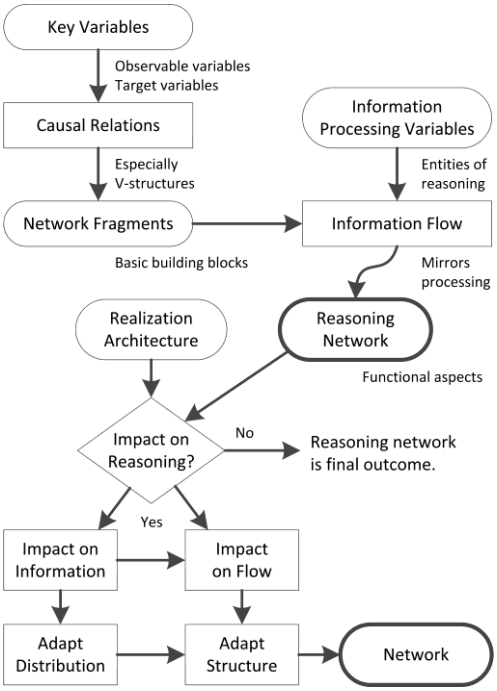


Fig. 5. Process of network generation

4 Experiments to Investigate Strategies

To compare the impact of the modeled decision strategies, e.g., the one for smart building with rooms without decision power and the one with smart rooms but no decision power for the building, we conduct a set of experiments. In each of those, applicable environmental settings, like *building usage*, are set (entered as evidence) in the respective Bayesian network (one per strategy). The effects of all possible energy saving policies or other circumstances on target variables like *discomfort* and *consumption* are computed individually via the probabilistic inference of the network, i.e., the circumstance is set and the Bayes network infers the distribution of all target variables given the evidence entered for the environmental settings.

This setup allows for experiments on the sum of all environmental circumstances, but also investigations that are specific, e.g., for high occupancy under extreme weather conditions and demanding policies. While a presentation of our actual results is pointless within this presentation of the approach due to the simplification of the networks, we present some details that we consider useful to gauge the approach. With regard to the efforts, we note first that an in-depth comparison of two strategies is possible within a few hours, while a high level estimation is merely a matter of minutes once the modeling is done. Second, we advise to run additional experiments with the purpose to test the models, establishing a standard engineering practice to

safeguard results and thus subsequent business decisions. Mostly, such test will look into fragments and their links, but also wider aspects are sensible. In our example, we checked that the target variables *consumption* and *discomfort* show identical values for all comparable experiments if no energy saving policy is in effect, but we also required that a smart building can leverage its information advantage, i.e., mere smart rooms should not outperform it on these indicators.

In addition to these investigations, it is possible to reason backwards, from effects to possible causes, e.g., to check the probability of circumstances that lead to levels of discomfort that are unacceptable. This is relevant for the dimensioning of systems and the setting of strategies, as it mirrors service-level agreements. Another type of experiments is the investigation of so-called counter-factuals that describe an alternative reality, e.g., a different building in which the condition of the room leads to different needs of those rooms, e.g., due to stronger insulation. This is useful to initiate change and improvements in a sensible manner. In [2], we detail the workflow for all types of experiments and provide insights into the stability of the results inferred from probabilistic summaries w.r.t. the precision of the network parameters.

5 Operational Use

Once a suitable decision strategy is found with the analysis methods described here, it is possible to implement it with live reasoning that uses a Bayesian network which takes observational data into account. The Bayesian network used for analysis forms a direct input for this; it is sufficient to adapt the level of details to the one observable for the building automation, keeping the structure of the reasoning intact. This works due to the construction of the network as system summary – analysis and operational control use identical reasoning.

6 Discussion

We introduced a method to model decision architectures of smart systems-of-systems that take their context, configuration, and current situation into account to change their behavior according to a dynamic or pre-set strategy. Our modeling summarizes the system, its behavior, and the impact of control decisions. This summarization is encoded in Bayesian networks; it uses probabilistic distributions over all key variables and their relationships. Using such a probabilistic summary allows to investigate architectural alternatives regarding system-level control strategies and the intelligent management of scenarios. This can be done with little efforts in experiments in which effects – and thus costs and benefits – of alternatives are computed and compared. Given the light-weight and modular manner of modeling, this sequence of modeling, experimental analysis, and investigation of cause-and-effect relations grants the benefits of an exploration of the design-space of the system's control architecture and behavior that is based on computational science. This allows system architects to take sound decisions, e.g., on where to put the 'smarts' of a smart building.

6.1 Relation to Engineering and AI

Our work addresses the architecting of smart behavior and control structures of smart systems-of-systems. It is therefore at the interface of system-of-systems engineering to artificial intelligence; two communities that only recently started to interact. It is our observation that existing work of these domains often take the contribution from the other domain out of their considerations: AI puts forward algorithms for smart behavior given a system layout – system engineering investigates designs assuming fixed behavior patterns (which are often close to traditional engineering). We believe that these more isolated points-of-view disregard the impact of a system's architecture to the possibility, quality, and efficiency of computations for behavior and control, and, vice versa, the demands of such computations on architecture. Realizing that the 'when' and 'where' of computations affects the 'what', we identified the need for advances w.r.t. the architecting of behavior and system-level-control, for which we propose our *probabilistic system summaries* as one modeling and analysis technique.

While we see our main contribution in this focus on a smart system-of-systems' architecture, we advise a strong link to operational aspects of decision strategies for future work. Medina-Oliva et al., e.g., propose to support the assessment of maintenance strategies of industrial systems using probabilistic relational models (PRM) in [10]. Their use of key performance indicators as optimization goals within a probabilistic framework is similar to our use of utility variables, pointing to a feasible integration of this work with sustainable operations management [11].

6.2 Implementation and Feasibility

The efforts to conduct an architecture investigation with *probabilistic system summaries* is very low, especially in comparison to alternatives like a detailed simulation where all individual objects are modeled. This assessment is based on experience, as we cannot pursue alternative approaches for various techniques in detail. We can, however, pinpoint various additional advantages: First, the models are Bayesian networks, for which both commercial and open-source software tools exist that offer well-established algorithms and suitable human-computer-interfaces. There is no need for additional engineering tools. Second, the modeling is dual purpose in the sense that models for analysis may be used for the operation of the smart system as well – thus reducing engineering efforts. Given the capabilities of smart systems, building automation systems in our domain, and the efficiency as well as real-time suitability of Bayesian networks, it is feasible to realize this with little to no extra costs.

However, we must re-consider the modelling efforts together with the efforts to fuse network fragments when we extend our approach to include more aspects into the architectural summary, e.g., for operations management as proposed above. Work by Koller et al. on probabilistic relational models [12] together with the foundations on probabilistic frame-based systems in [13] covers the modelling of large complex domains with the coherent probabilistic representation of Bayesian networks. As this work has many application domains, we expect further advances regarding tool support, further guaranteeing the feasibility of industrial use of modelling system summaries.

Acknowledgements

This work was partly funded via the EU FP7 ICT project SCUBA – Self-organising, Co-operative and robust Building Automation (www.ict-scuba.eu).

References

1. Scuba FP7 Project, Scientific papers. www.aws.cit.ie/scuba/index.php/documents/scientific-papers/
2. Borth, M.: Reasoning for smart control strategies. In Scuba FP7 Project, Deliverable 4.2: Scenario management, control and self-organization strategies, initial adaptation strategies, in press (2014)
3. Jensen, F.V.: Bayesian networks and decision graphs. Springer (2001)
4. Pearl, J.: Causality. Models, reasoning, and inference. Cambridge University Press (2000)
5. Jensen, F.V., Aldenryd, S.H., Jensen, K.B.: Sensitivity analysis in Bayesian networks. In: Symbolic and Quantitative Approaches to Reasoning and Uncertainty Vol. 946, Carbonell, J.G., Siekmann, J., Goos, G., Hartmanis, J., Leeuwen, J., Froidevaux, C., Kohlas, J. (eds.) Springer Lecture Notes in Computer Science, pp. 243–250 (1995)
6. Koller, D., Pfeffer, A.: Object-oriented Bayesian networks. In: Geiger, D., Shenoy, P.P. (eds.) Proceedings of the 13th conference on uncertainty in artificial intelligence (UAI'97). Morgan Kaufmann Publishers Inc., pp. 302–313 (1997)
7. Laskey, K.B., Mahoney, S.M.: Network fragments: representing knowledge for constructing probabilistic models. In: Geiger, D., Shenoy, P.P. (eds.) Proceedings of the 13th conference on uncertainty in artificial intelligence (UAI'97). Morgan Kaufmann Publishers Inc., pp. 334–341 (1997)
8. Borth, M., von Hasseln, H.: Systematic generation of Bayesian networks from systems specifications. In Musen, M.A., Neumann, B., Studer, R. (eds.) Intelligent information processing. Kluver, pp. 155–166 (2002)
9. Kjærulff, U.B., Madsen, A.L.: Bayesian networks and influence diagrams: a guide to construction and analysis. Springer (2013)
10. Medina-Oliva, G., Weber, P., Iung, B.: PRM-based patterns for knowledge formalisation of industrial systems to support maintenance strategies assessment. Reliability Engineering & System Safety, 116, 38–56. doi:10.1016/j.ress.2013.02.026 (2013)
11. Kleindorfer, P.R., Singhal, K., Van Wassenhove, L.N.: Sustainable operations management. Production and Operations Management, vol. 14 (4) pp. 482-492 (2005)
12. Getoor, L., Friedman, N., Koller, D., Pfeffer, A., Taskar, B.: Probabilistic relational models. In: Getoor, L., Taskar, B. (eds.) An introduction to statistical relational learning, MIT Press (2007)
13. Koller, D., Pfeffer, A.: Probabilistic frame-based systems. In: Mostow, J., Rich, C., Buchanan, B. (eds.) Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence (AAAI '98/IAAI '98). American Association for Artificial Intelligence, pp. 580-587 (1998)

www.UniText.fr
Information System Concept for Documentation/Project Management

Philippe JARRIN[1], Luc BEAUPERE[1]

[1] www.LesAmisDeUniText.fr,
www.UniText.fr
Philippe.Jarrin@Wanadoo.fr
Luc.Beaupere@Magic.fr

Abstract. The UniText tool is a concept in development that takes into account the distributed nature of documentation, which consists of multiple documents linked to several parts of the system described at different levels of abstraction, and produced by different people with different authority levels. UniText can manage the traceability of all the documentation units according to several criteria (author, date, context, tasks, sub-projects, delegation of projects), as well as suggestions and discussions about the documentation, in order to help building and maintaining a consistent documentation from all these pieces of information. The UniText venture is designed to structure the work of managers who are responsible for documentation in R&D projects, in order to improve clarity, motivation, efficiency, capabilities and to absorb and respect all contributions of all stakeholders.

1 Introduction

Documentation is a key issue in complex projects and it is often a very time-consuming activity which has a low perceived value, when the same units of text must be repeated over and over in multiple documents, correlated differently, with several levels of synthesis and/or analysis and aimed at many stakeholders points-of-view. Out-of-date, misdelivered and/or uncorrelated documents lead to unnecessary mistakes and configuration issues with the consequence, among several, being extra costs and delays due to partial re-working.

Documentation management is "local" and "asynchronous" -- when the reference documents are part of costly contracts within an "extended enterprise" -- and is also subject to "change management" -- thus creating added requirements for updating peripheral documents needed for contractual changes, negotiations, decisions, new iterations for handling an ever-evolving flow of data, etc. In response to the need for documentation management stream-lining, **UniText** has been developed as an integrated documentation management system, based on small units of text, linked in a

network of contexts, locally organized in hierarchy, and automatically managed in configuration by date.

This software will link colleagues, their authority, the propagation of access rights, including delegation of projects, sub-projects and incidental tasks. Ideas will be managed in a network of context and date, and will have discussion attributes at different levels including offers of anonymous suggestions (whose contribution will carry no weight of commitment), straightforward decision-making (carrying the weight of commitment of a set of signers) and descriptions of work (carrying the weight of commitment of a set of signers who can describe the existing system).

UniText is a complement to the management/evolution of documents, due to tremendous reduction in the number of updates and iterations made to individual documents removed from a particular project's stream of documentation management. Within **UniText** , ideas are managed in configuration with virtual contents that can be manipulated by several users. A dynamic project management workflow system enables vibrant discussion, clear and shared decision-making. In this way, **UniText** will create a virtual think-tank environment to consolidate the set of applicable documents.

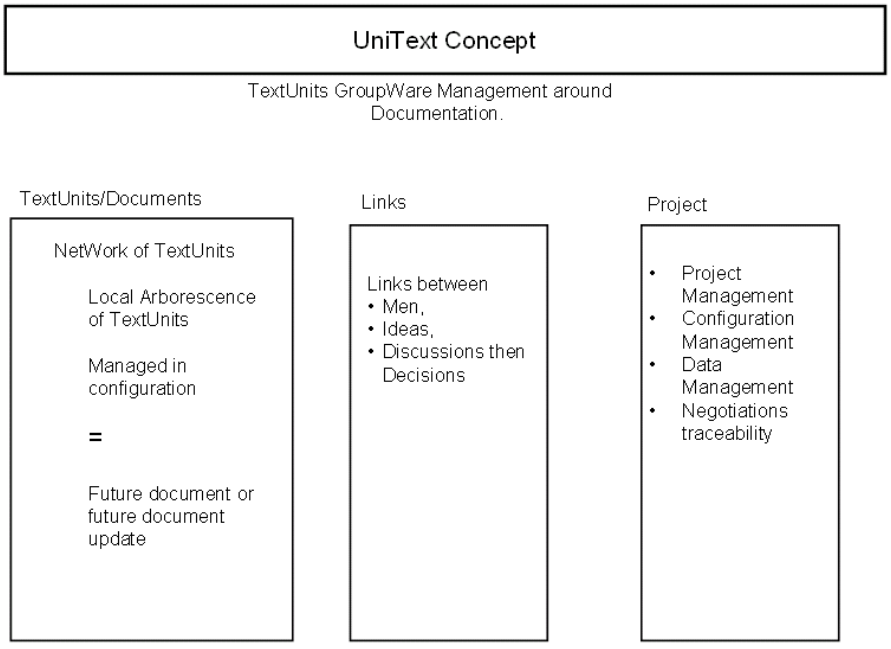


Fig. 1. UniText Concept Overview.

2 Existing tools and methodologies

The UniText concept proposes a compact intranet/internet-based environment wherein "text units" are managed in configuration with a project's global set of data in order to produce inter-related documents. Text units are units of text, as in short sentences, for example, chosen for their applicability within requirement management which are published and shared amongst a project's stakeholders.

Emails, for example, are text units which are not currently capitalized upon in order to produce final documents with any configuration management structure. The usage of text units is common on the web, but this capability has not been exploited yet to produce complex documentation in systems design. Management of "document-like" pages is also common in Wiki pages. Project management standards are also well described, as stated in the *Reference* section of this paper, and the UniText goal is to implement software solutions to comply with these environments and standards. The originality of UniText is to create a new usage of distributed text units, from a synthesis effect, now possible thanks to mature means available through intranets and the internet.

The UniText concept is not a revolution in tools or methodologies. Let's look at the common process to negotiate documents in systems R&D :

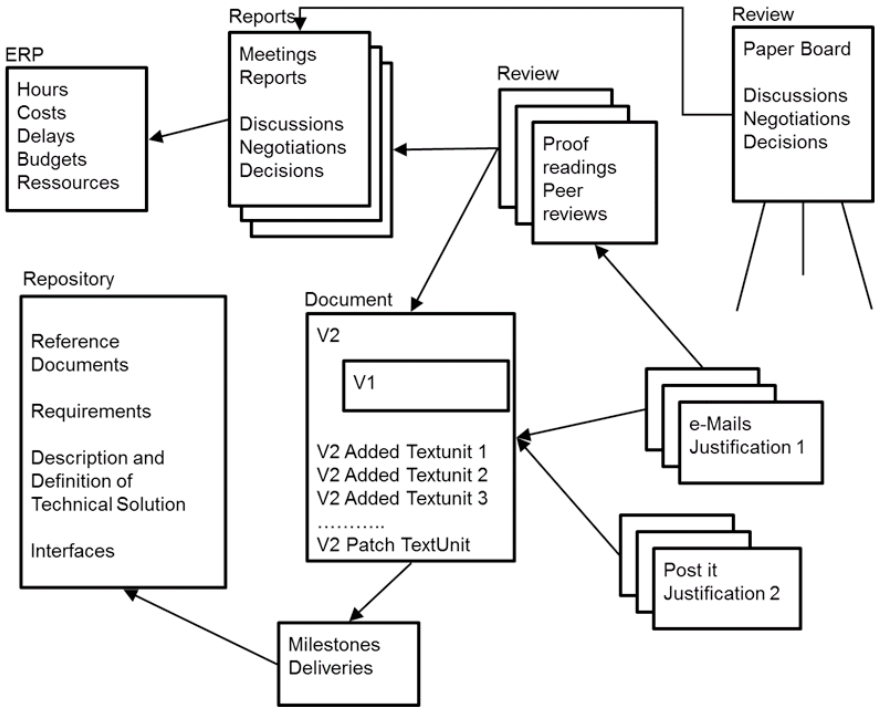


Fig. 2. Usual process to elaborate and discuss documents.

3 Issues addressed in documentation update

As we can see in Figure 3 below, many documents have to be updated from version *n* or *Vn* to the next version, *n*+1 or *Vn*+1.

In large and complex projects, a lot of subtle information, complexity and quantity of communication are lost. As shown above in Figure 2 it is not possible to communicate well outside of a "core team", or a reduced set of people responsible for a part of the project/product/service with limited and manageable interface with the outside world.

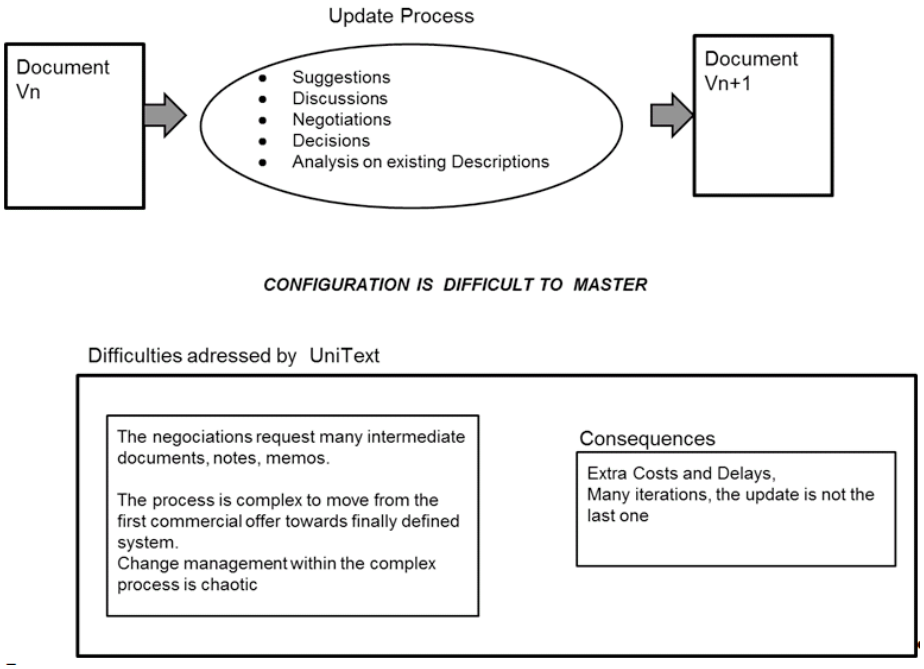


Fig. 3. Issues addressed

The main issue considering cost and delays is the difficulty to predict how many iterations will be needed to obtain a final set of documentation satisfactory to all stakeholders. We generally know the goal, which final documentation is needed, but how to go there requires iterations.

4 Organization of the UniText concept.

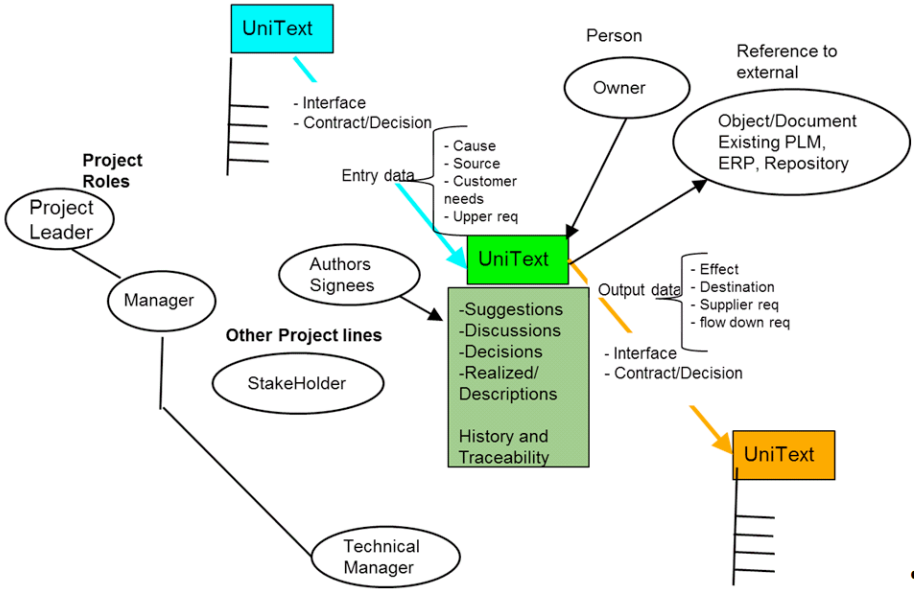


Fig. 4. Organization of the UniText concept

The UniText concept is centered around the person, and around a network of ideas, globally in the network, and locally in a hierarchy. The existing intranet/extranet technology enables the different interactions required for forging the efficient links between men, decisions and ideas. UniText's internal database is meant to be local and asynchronous, centered on the person. As a person, do I need to know the global OBS of the project, the global organization of my customers, of my company, of my suppliers? UniText's local philosophy negotiates the inter-dependency between people at the individual level, the one, two or three stakeholders who will back me up as a project leader or a hierarchic. The local data model of UniText describes links between people, with propagation rules: What delegation do I receive? What delegation can I propagate? This relationship between stakeholders and data propagates access rights, ordering what access rights to which project they receive and what access rights they can propagate. In complex projects, the repository of files and documents generally requires a system administrator to permit access to different shared hard disk directories.

The ambition of UniText is to declare a bottom-up logic, and to enable access rights to stakeholders as naturally as one can give a document to a colleague from hand-to-hand. The goal is to extend a lot of freedom to contribute to the compilation of documentation to all stakeholders. All ideas and suggestions would find or create a

correct context to be written in. The result would be the motivation and respect of stakeholders who will have access to an easy way to communicate their concerns in a transverse way to the right people, at the right time.

5 Conclusion

The UniText project is a free software concept to be used within extended enterprises, from clients in-charge of CSDM to final detailed-parts suppliers, managing documentation, related to projects. This software is an attempt to organize locally, at the individual stakeholder level, the best management practices in high-tech industries like aeronautics, where complex documentation is required and has to be maintained for financial, contractual, reliability, long-term maintainability, traceability and safety reasons. The goal of the association *Les Amis de UniText* is to develop an initial software model within DRUPAL. DRUPAL will enable security and web compatibility. The initial software data model will contain nine tables and a total of around ninety fields.

Your input and contributions of ideas, programming, testing and financial donation are welcome.

References

1. Philippe Jarrin – Blog – www.UniText.fr 2013-2014
2. Luc Beaupère - www. LesAmisDeUniText.fr
3. FD ISO 10005 - September 2005 : Guidelines for quality plans
4. FD ISO 10006 - December 2003 : Guidelines for quality management in projects
5. NF X50-151, NF EN 16271 - Février/February 2013 – Value Management (Industry)
6. Project Management Body of Knowledge (PMBOK® Guide) —Fourth Edition
7. ISO 10007 Second edition - 2003-06-15 : Configuration Management

Correct by Prognosis: Methodology for a Contract-based Refinement of Evolution Models *

Christoph Etzien and Tayfun Gezgin

OFFIS, Escherweg 2,
26121 Oldenburg, Germany,
{christoph.etzien,tayfun.gezgin}@offis.de

Abstract. The scope of this paper is collaborative, distributed safety critical systems which build up a larger scale system of systems (SoS). Systems in this context are independently designed and can operate autonomously following both global SoS goals and individual goals. A major aspect of SoSs is the evolution over time, i.e. the change of its architecture as a result of changes in the context of the SoS or the changes of individual or global goals. The aim of this paper is to define a modeling concept for evolution specifying all possible changes of the SoS over time. This evolution model is used to generate and analyze future architectures enabling the prediction of future violations of static specifications. We derive so called *dynamicity contracts* and restrict the evolution model in such a manner, that *false* architectures are not reached.

1 Introduction

In recent years the co-operations and inter-connections between individual, geographically distributed systems heavily increased, leading to a new paradigm called *Systems of Systems* (SoSs). Also in safety critical areas the significance of these topics increased. As an example, much effort has been invested in the development of *Car-to-Car* communications with the aim to increase the safety in traffic and optimize traffic flows. Another example is the dynamic partitioning of the airspace with respect to time investigated in the SESAR (Single European Sky ATM Research) program. The recent partitioning of the airspace is performed in a statical manner with respect to time, i.e. the trajectories are not changed during the whole landing approach and the take off. The shift to a *dynamic* partitioning, which is called 4D-trajectories, involves a much more intensive co-operation between the tower and each airplane.

To distinguish between complex systems and SoSs, Mark Maier defined a set of characteristics [1], like the *geographical distribution* or the *operational independence*. The more a system exhibits these characteristics, the more it is an

* This work was supported in part by European Commission for funding the Large-scale integrating project (IP) proposal under the ICT Call 7 (FP7-ICT-2011-7) "Designing for Adaptability and evolutioN in System of systems Engineering (DANSE)" (No. 287716).

SoS. The main characteristic we are interest in is the *evolutionary development*, i.e. the change of the architecture of an SoS during its lifetime. A model for the evolutionary development can be created based on prognosis on possible future evolutions of the SoS. As an example, statistical data could be used to do a prognosis on the future traffic density in a district of a city. We propose *graph grammars* to model the possible evolutions of an SoS. These transformations could also be specified via temporal logics as we proposed in [2]. However, the specification with graph grammars is more intuitive than temporal logic.

Graph grammars describe the adaption to a context change in form of transformation rules. With these transformation rules the inter-connections of the constituent systems and thereby their roles and interaction protocols are changed. The trigger of such transformation rules are the constituent systems itself: When systems adapt or change their local goals and thus affect their local behavior, change their services offered to the environment, or need some services from their local environment, a request to change some parts of the SoS architecture are triggered from the corresponding constituent systems. In [2] we already discussed the initiations for transformation rules from constituent systems.

Besides the evolution model, we will consider static specifications of SoSs by contracts defining invariants and constraints on the architecture of the SoS. An example for a static contract is that systems applying inconsistent roles should not co-operate. The set of static contracts of an SoS defines all legal architectures of this SoS. Beginning with an initial SoS architecture, the evolution model successively generates a set of successor architectures. Transformations are applied locally resulting in sequences of transformations which could lead to an architecture violating the static contracts. In this paper, we derive so called *dynamicity contracts* which restrict the dynamics of the evolution model of the SoS to prevent the SoS entering an architecture which violates its static specification. We extend this approach by tolerating a finite set of intermediate architectures, which violate the static specifications. These intermediate architectures have to be left finally and a *safe* architecture has to be reached within a specified number of changes. The idea to allow temporarily intermediate faulty architectures is inspired by the *fault tolerance time intervals* defined in the ISO 26262 [3]. After the occurrence of a fault, a safe system state has to be reached within a defined time interval. If this interval exceeded, an hazardous event could occur.

To model the static architecural part of an SoS we use the UPDM framework [4]. UPDM is a unified Profile for DoDAF (Department of Defence Architecture Framework) and MODAF (Ministry of Defence Architectural Framework). It supports the capabilities to model architectures of complex systems, system of systems, and service oriented architectures. Beside milestones, no dynamicity aspects of systems of systems were considered in this framework.

1.1 Related work

In [5] a method for modeling and analyzing the dynamicity for multi-hop ad hoc networks was presented. Statistical estimation theory was applied to model the so called *configuration* of a multi-hop wireless network. In [6] a supporting model

called *dynamicity aware graph relabeling system* is introduced. This model is used for ad-hoc networks to take mobility into account. Ultra large scale systems are the topic of [7], where the main characteristics are captured and specified, e.g. decentralized control, conflicting requirements, and continuous evolution. In [8] some major issues in self-coordinating systems are depicted. The main statement is that a tight integration of all disciplines in the development process of such large scale self-coordinating systems has to be established. An approach for the design and analysis of multi-agent systems was presented in [9]. Agents are able to sense and manipulate specific aspects of the environment. Sets of agents form community types, which interact in the modeled environment with some interaction specifications. In [10] self-adaptive systems were presented. Initially a system architecture with defined components, their interfaces, and a set of coordination pattern is given. Coordination pattern define protocols between components via roles. Reconfigurations are defined via graph transformation rules and are initiated by environmental changes.

Automatic verification of the real-time behavior including the reconfiguration is supported by CHARON [11], Masaccio [12], and *Mechatronic UML* [13]. There are some approaches for modeling the structural aspects of adaptive systems [14, 15] or the behavioral aspects [16, 17] but none of them consider both aspects.

1.2 Outline

The following section introduces the fundamentals of our work, i.e. the considered modeling formalism called UPDM, the contract-based specification formalism, and the formalisms needed to express transformations. Section 3 illustrates our approach to derivate dynamicity contracts in order to prevent the SoS to evolve in architectures which violate its static specification. In Section 4 we illustrate our implementation and give some example scenarios. Finally, Section 5 concludes the paper and discusses some further work.

2 Fundamentals

The basic modeling elements of our approach are components as structural elements, and graph grammars. Our components are enriched by so called contracts, specifying the allowed context of a component and its guaranteed behaviour. Components and contracts are detailed in the following section.

2.1 Contract-based Modelling

We use *Heterogenous Rich Components* (HRCs) [18, 19] to model systems and its artifacts in a black box manner. The dynamics of an HRC can be specified by, e.g., an external behavior model. For each HRC a set of specifications in terms of contracts [20] is defined. A contract is a pair consisting of an assumption (A) and a guarantee (G). The assumption specifies how the context of the component, i.e. the environment from the point of view of the component, should behave. Only if the assumption holds, then the component will behave as guaranteed.

The system decomposition can be verified with respect to contracts without the knowledge of the concrete implementation. The specification of both assumptions and guarantees can be provided based on a pattern based language like introduced in [21].

Having a formal specification for the component and its sub-components the so called *Virtual Integration Test* (VIT) [22] can be performed. It is called virtual because no implementation for the sub-components or any testbed is required. This analysis is performed based on the specifications, the interfaces, the connections, and the structure of the composition. This test checks if the composition of the sub-component contracts implies the contracts of the surrounding component. In this work we assume that the components are implemented according to their contracts and call an architecture *valid* iff the VIT is successful.

2.2 Rewriting Rules

An *architecture* of an SoS is a composition of CSs at a specific time, where *roles* and inter-connections of all systems are specified. Changes of an architecture of the SoS are defined by a set of rewriting rules. Rewriting rules consist of a left hand side and a right hand side corresponding to architectures of the SoS. In this work, rewriting rules restructure the architecture of an SoS by composing or separating system instances in a well-defined way, and applying the right roles to the corresponding systems. So, a single transformation affects a subset of participating system instances, their inter-connections, roles, and modes. In the following, we will formalize the concept of graphs and rewriting rules.

A graph is defined as a tuple $G = (V, E, s, t)$ where V is a set of vertices, E is a set of edges and s, t are a source and a target function defined as $\{s, t\} : E \to V$. Let L, R be two graphs. A rewriting rule $r : L \to R$ is defined in such a way, that whenever an instance of L, called *match*, is found in a graph G, this instance can be replaced through an instance of R leading to the transformed graph G'. For two graphs H, G let $h : H \to G$ be a *graph homomorphism*, mapping nodes and edges of H to G. The homomorphism consists of two functions $h_V : V_H \to V_G$ and $h_E : E_H \to E_G$, such that $\mu_G \circ h_E = h_V \circ \mu_H$, with $\mu = \{s, t\}$. A rule $r : L \to R$ can be applied to a graph G leading to a changed graph G', in short $d : G \to_r G'$, if there exist two homomorphisms h_1, h_2, such that $h_1 : L \to G$ and $h_2 : R \to G'$. In Section 3 we will apply a set of rewriting rules specifying the dynamic behaviour of an SoS.

2.3 Modeling the SoS

System of systems (SoS) consist of several constituent systems (CS) which are instances of systems or even SoSs themselves. To distinguish between complex systems and SoSs Maier proposed five criteria for the "SoS-ness" of a complex system, which are introduced in the following [1]:

- *Operational independence of elements*: The CSs can operate independently.
- *Managerial independence of the elements*: The CSs are separately acquired by different managerial entities.

- *Evolutionary development*: An SoS evolves over time, developing its capabilities as the CSs are changed, added, or removed.
- *Emergent behavior*: The SoS itself offers additional services beyond the capabilities of the CSs including unexpected and potentially damaging behaviors.
- *Geographic distribution*: The geographical extent of the CSs could be "large".

We focus on the *evolutionary development* aspect of SoS and therefore concentrate not only on the architecture at a specific time but also on the evolution of the CSs and their re-configurations. We distinguish two levels of behavior, i.e. *system dynamics*, and *evolution*. System dynamics deal with the question, how systems exchange data via their inter-connections. The topic *evolution* poses the question, how systems and their inter-connections are changed over time.

System dynamics are covered by the UML/SysML behavioral models and diagrams, e.g. state charts. We use contracts to specify the assumed and guaranteed behavior of each CS.

We address the evolutionary development and extend the milestone-based representation of SoSs in UPDM. A milestone represents an architecture of the SoS at a specific point in time. We will focus on the system view which basically represents *systems* itself, their *resource roles* and inter-connections. The SV-1 allows to characterize the inter-connections of the CSs for a single architecture. The milestone plan (AV-2) is the planed evolution of the SoS taking the entire life-cycle of the CS into account. This plan is created manually and explicitly since each milestone consists of an entire SoS architecture.

The problem we address is that the owners or managers of the CSs follow their own goals, and change or influence changes of their CSs independently from a central authority. We define *goal* as an optimization metric which represents how good (or bad) a CS (or an SoS) performs. These values can be statically computed for an architecture of the SoS, or depend on the system dynamics and are measured during execution. The owners of a CS are assumed to monitor this values and decide to change the behavior or connectivity of their CS to improve their goals. This change might take place on the CS level by switching into another mode or on the SoS level by changing the inter-connections to other CSs. In the first case this behavior is part of the CS specification and covered by the contracts of the CS. In the second case the change is beyond the system borders of the CS and therefore not in the scope of the specification of the system dynamics. The evolution behavior of the SoS is based on changes which might impact the dynamics of the SoS.

3 From Evolution Model to Contracts

A model for an SoS consists of an initial architecture, a static specification and an evolution behaviour. As said before, evolution behaviors define the possibility of re-configuring a given architecture as a result of e. g. changing environmental conditions, or some adaption of cooperations between a set of systems. We will apply graph grammars to model such a behaviour. The benefit of the usage

of evolution models is the possibility to generate and analyze future architectures enabling the prediction of future violations of specifications. Changes of a given SoS can be explored before they occur in *reality* in order to prevent invalid architectures of the SoS. Typically an infinite number of architectures will be generated by graph grammars. In our concept, we will apply the concept of bounded model-checking, i.e. we will only consider a finite number of architectures reached by a grammar specification. This also has a practical relevance, as in general the evolution model shall only predict the possible behaviour for a finite time frame instead of an infinite time frame. To obtain a finite set of architectures, we could apply abstraction techniques like the *Partner Abstraction* introduced in [23].

3.1 Derivation of Evolution Contracts

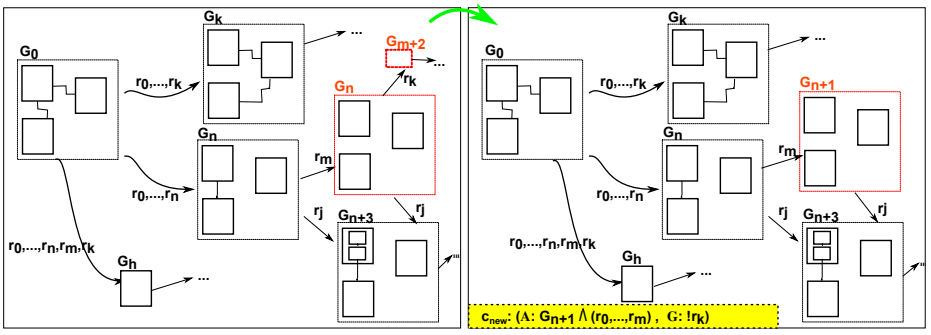


Fig. 1: Example scenario for derivation of evolution contracts – Left: Initial situation for a given SoS model; Right: Derivation of new dynamicity contracts.

The concept of dynamicity contracts complements the static contracts for the SoS and constituent systems. The static contracts restrict the allowed behaviour of the overall SoS and each system, whereas the dynamicity contracts restrict the dynamics of the evolution model of the SoS.

Starting from an initial architecture each reached architecture is analyzed if it is valid. If invalid architectures are reached, a dynamicity contract is derived in such a way, that the evolution model is prevented to generate this architecture. Thereby, the assumption part of a dynamicity contract encapsulates the architecture, from which a violating one can be reached by the application of a rule defined in the evolution model. The guarantee part then consists of the negation of the corresponding identifier of the rule. Further, we extend this approach by allowing intermediate architectures, which violate the static specifications, if a valid architecture is reached after "some time". In this work, we require that a specified *number* of successive invalid architectures may be tolerated, and after this number a valid architecture has to be reached. In future work, we will extend this approach by specifying some allowed *time frames*.

Consider the example of Figure 1, where the initial architecture G_0 can evolve to different future architectures by applying an evolution model consisting of a set of rewriting rules r. Assume that we allow that during the evolution maximal a single architecture may be reached which violates the static contracts. On the left part of the figure the initial situation is depicted, where no restrictions exist so far for our evolution model. If the sequence of rules $r_0, ...r_n, r_m$ is applied, we can reach the architecture G_{n+1} which violates the static specification. If we would now apply rule r_k we would again get an architecture violating the static specification. In order to restrict our evolution model we derive the contract illustrated in the right part of Figure 1. The contract states, that whenever we are in an architecture isomorph to G_{n+1} and we previously applied the sequence of rules $r_0, ..., r_m$, the rule r_k will not be applied. Note, that we need the architecture within the assumption part, as rewriting rules are non-deterministic. The sequence of rules $r_0, ..., r_m$ can also lead to some architectures not violating the static contracts as illustrated in Figure 1.

Next, we define our applied graph grammar formalism, and formalize the derivation of dynamicity contracts.

Graph Grammars Let $w = r_0, ..., r_n, ...$ be a word over an alphabet Σ, $pre(w, n) = r_0, ..., r_n$ be its prefix consisting of $n+1$ symbols, and $w(n) = r_n \in \Sigma$ the $(n+1)-$th symbol. A *dynamicity contract* is a contract talking about graphs and prefixed of words: The assumption (`A`) part of a dynamicity contract c consists of a (possibly empty) finite prefix of a word w and a graph G, its guarantee (`G`) consists of a symbol in Σ, in short $c:$ $(\texttt{A}: pre(w, i-1) \wedge G_i,\ \texttt{G}: !\sigma)$ for some $i \in \mathbb{N}$ with $\sigma \in \Sigma$. The intuition is that whenever a finite sequence of symbols $pre(w, i-1)$ is received and the graph G_i is reached, the next symbol shall not be σ. With these dynamicity contracts we will restrict graph grammars in such a way, that through the successive application of rules it always holds, that no graph can be reached violating some static specifications.

A graph grammar is a tuple $\mathcal{G} = (G_0, R, C_D)$ where G_0 is a start graph, $R = \{r_0, ..., r_k\}$ is the set of rewriting rules (each with an unique identifier), and C_D is the set of dynamicity contracts, which may be empty at design time. In the next section we detail the iterative extension of this set. A graph grammar can be translated to a finite $\omega-$automaton $T_E = (S, s_o, \Sigma, \rightarrow)$, where S is a set of graphs corresponding to the set of states, s_0 the initial state, Σ an alphabet consisting of the identifiers of the rules in R, and $\rightarrow \subseteq S \times \Sigma \times S$ the transition relation. All states are considered to be accepting ones.

A run ρ of T_E over an infinite word $w = r_0, ..., r_n, ...$ is an infinite sequence of graphs $G_0 \rightarrow_{r_0} ... \rightarrow_{r_n} G_n, \rightarrow_{r_{n+1}} ...$ such that G_0 is the initial graph and $(G_i, r_i, G_{i+1}) \in \rightarrow$ for all $i, j \in \mathbb{N}$, for which holds that $(\texttt{A}: pre(w, i-1) \wedge G_i,\ \texttt{G}: !w(i)) \notin C_D$. The language of a graph grammar is defined as the set of words accepted by its finite automaton.

Derivation of Dynamicity Contracts A specification for an SoS is given by the tuple $SoS = (\mathcal{G}, C_s)$ where $\mathcal{G}$ is a graph grammar specifying the evolution model, and C_s a set of static contracts defining the allowed SoS behaviour. In

general, the evolution model specified through the concept of graph grammars is not initially consistent with the static specification specified as a set of contracts, as rewriting rules are applied locally resulting in sequences of rules which could lead to a graph violating the static contracts. This can happen because the application of a rule does not check whether the reached graph harms a static contract. In order to make the evolution model consistent with respect to the contract specification, such paths have to be removed from the evolution model. For this we derive new dynamicity contracts from these paths.

The easiest case is given, when a direct application of a rule violates a static contract and no intermediate architectures violating contracts are allowed. For such cases we can derive a dynamicity contract consisting of the current architecture G as the assumption part, and the negation of the identifier of the corresponding rule for the guarantee part. That is, we extend our dynamicity contract set C_D of $\mathcal{G}$ with the contract $\{(\mathtt{A}:\ G,\ \mathtt{G}:\ \sigma)\}$, if there exists a rule $\sigma : L \to R$ in $\mathcal{G}$, and $G \to_\sigma G'$ could be applied, such that $G' \not\models C_s$.

With this extension the graph grammar will be prevented by firing rule σ when an isomorphic graph to G is present. If violating graphs are accepted temporarily, e.g. a finite amount of time, or a finite number of violating graphs, we need to extend such dynamicity contracts with the history which lead to a corresponding architecture. In this work, we will only consider the maximal successive number of incorrect intermediate architectures, i.e. architectures violating the static specification.

Let $\xi \in \mathbb{N}$ be the maximal number of successive graphs violating the static specification, which is defined to be tolerable. Let $pre(w, n) = r_0, ..., r_n$ be a prefix of a word, for which there exists a run $\rho = G_0 \to_{r_0} ... \to_{r_i} G_i ... \to_{r_n} G_n$ of the automaton of $\mathcal{G}$, such that $G_i, ..., G_n \not\models C_s$ and $|\{G_i, ..., G_n\}| > \xi$. Then we derive the following dynamicity contract and extend the set C_D as follows:

$$C_D \cup \{(\mathtt{A}:\ G_{n-1} \wedge pre(w, n-1),\ \mathtt{G}:\ !w(n))\}. \tag{1}$$

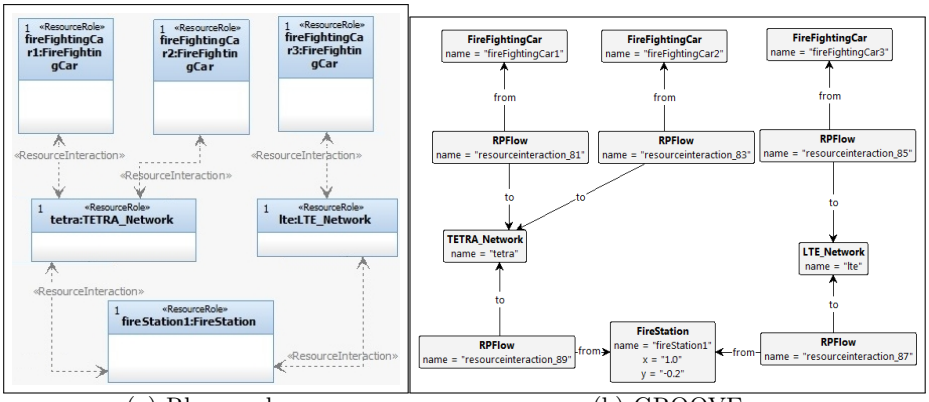
Note that a word w could result in a set of runs instead a single run. In this case our new dynamicity contracts are *correct* in the sense, that no *legal* evolutions resulting in graphs which all fulfill the static contracts are excluded. This is because the assumption part exactly states, that a rule shall not be applied if a specific architecture is given.

4 Application of Methodology

To illustrate our approach we consider an emergency response scenario, consisting of a set of constituent systems like fire stations and fire brigades. All CSs participating in this SoS shall behave cooperative in order to minimize the needed time for an operation in case of an emergency.

We use a new custom diagram via an additional profile which allows to model rewriting rules graphically in IBM Rational Rhapsody©. These diagrams allow to add placeholders which refer to model elements of the Rhapsody UPDM model. This reference mechanism ensures that the model itself and the rewriting

rules are clearly separated. The rules contain four different kinds of graphical elements for each CSs and their inter-connections, i.e. *Reader*, *Creator*, *Eraser* and *Embargo*. *Reader* elements are unchanged elements of a corresponding rule. *Creator* elements represent newly generated elements on the right hand side of the rule. *Eraser* elements address elements of the left hand side which are removed via the rule application. *Embargo* elements restrict the applicability of the rule if the match can be extended by these elements. The Rhapsody model including its rules are exported to GXL[24] files which are the input language of the GROOVE[25] tool. GROOVE is used for the generation of architecture alternatives and is also able to perform the isomorphism check of the generated architectures. After applying GROOVE we get a set of architectures, and the corresponding network representing the applied rules. As an example consider

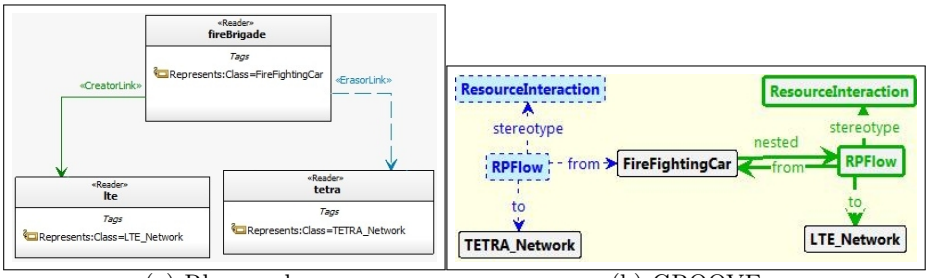


(a) Rhaposody (b) GROOVE

Fig. 2: Excerpt of the Emergency Response System

Figure 2 and 3. The purpose of the fire service is to delete fire at any location within a city and to save the involved people. The time between the incident harms people and the treatment begins is critical for the recovery of the injured. Therefore the goal of the fire service is to minimize the time between the notification and the arrival of the right amount of units to treat the injured people at the incident location. Increasing traffic density typically extends this time frame and might require to send units from locations with a larger geographical distance but lower distance in travel time. To improve this, one option is to increase the number of units like fire brigades but this is only partially possible. Another option is to increase the awareness of the fire head quarter about the required number (and kind) of units at the location. This can be achieved by improving the communication technology, in this scenario the change from the current TETRA [1]) to the LTE[2]) communication technology. The application of such a rule leading to an architectural change is illustrated in Figure 3(b). In

[1] TETRA: *Terrestrial Trunked Radio*, ETSI EN 300 392-2 v3.2.1
[2] LTE: *Long-Term Evolution*



(a) Rhaposody (b) GROOVE

Fig. 3: Rule Translation: Rules in Rhapsody (a) are automatically translated into rules in GROOVE (b)

this example, the evolution model contains only a very small set of architectures because the rule is only applicable once per fire brigade and the number of fire brigades is low. If one would add a rule adding fire brigades to the model the number of architectures would be infinite. In the complete model several fire stations are coordinated by one head quarter and also the number of fire brigades is higher. Since the fire brigades are coordinated by the different fire stations and must cooperate during operation it is essential that those brigades use the same communication network. If each brigade is updated to the new technology individually, invalid architectures are possible which can be characterized as (at least) two brigades coordinated by the same fire station using different networks. The evolution must be restricted to avoid those architectures. Typically not all those constraints can be derived from reasoning about architectural pattern only but the reachable architectures have to be analyzed including the system dynamics. This can be done via simulation or static analysis (e.g. timing analysis as proposed in [2]). The results of the analysis are annotated to the reachable architectures and support the identification of contracts for the evolution itself.

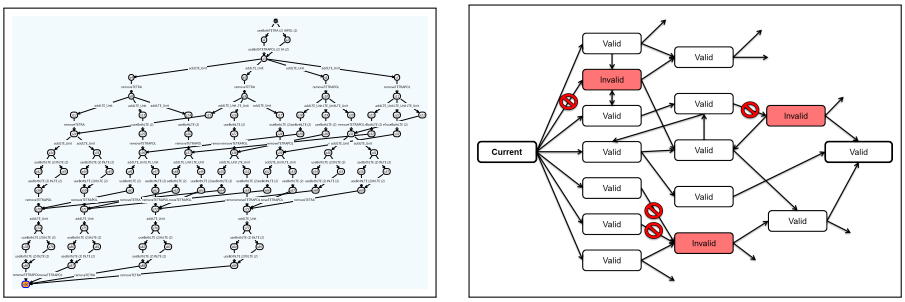


Fig. 4: Left: network of architecture alternatives; Right: annotated network.

In the left part of Figure 4 a network of reachable architectures is illustrated. In the right part of Figure 4 a (simplified) network of architectures is presented.

For this network the invalid architectures are marked in red. From this network global constraints are derived which restrict the application of rules. These conditions are the previous architectures of any edge ending in an invalid architecture. The evolution contract takes this condition as assumption and the negated invalid architecture as guarantee.

5 Conclusion

We presented a modeling concept for evolution specifying all possible changes of the SoS over time as an extension of the UPDM framework. We introduced a novel approach for deriving dynamicity contracts restricting such evolution models in order to prevent reaching invalid architectures with respect to the static specification of an SoS. Our prototype implementation offers so far an export mechanism from UPDM models created with Rhapsody to GROOVE, and feed back the generated architecture alternatives to Rhapsody. For the generated models we can apply our previously introduced virtual integration checker [26] and manually derive dynamicity contracts. Currently, we aim to close this loop, i.e. the generation of architectures and calling the verification back end to automatically generate dynamicity contracts. In future work we also plan to include the notion of time for the evolution models to enable reasoning about timing constraints for the evolution.

References

1. W.Maier, M.: Architecting principles for systems-of-systems. In: Inc. Systems Engineering. Volume 1. (1998) 267–284
2. Etzien, C., Gezgin, T., Fröschle, S., Henkler, S., Rettberg, A.: Contracts for evolving systems. In: SORT – The Fourth IEEE Workshop on Self-Organizing Real-Time Systems. (06 2013)
3. ISO26262: Road vehicles – functional safety (2011)
4. Group, O.M. In: Unified Profile for DoDAF and MODAF. (2008)
5. Hamlili, A., Morocco, R.: A common computational approach analyzing dynamicity and connectivity for reliable communications in multihop wireless networks. In: Int. Conf. on Models of Information and Communication Systems. ICST Alliance
6. Casteigts, A., Chaumette, S.: Dynamicity aware graph relabeling systems (da-grs), a local computation based model to describe manet algorithms. In: IASTED PDCS. In proceeding of: International Conference on Parallel and Distributed Computing Systems, PDCS, Phoenix, AZ, USA (2005) 231 – 236
7. Northrop, L., Feiler, P., Gabriel, R.P., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Schmidt, D., Sullivan, K., Wallnau, K.: Ultra-Large-Scale Systems - The Software Challenge of the Future. Technical report, Software Engineering Institute, Carnegie Mellon (June 2006)
8. Schäfer, W., Birattari, M., Blömer, J., Dorigo, M., Engels, G., O'Grady, R., Platzner, M., Rammig, F., Reif, W., Trächtler, A.: Engineering self-coordinating software intensive systems. In: Proceedings of the Foundations of Software Engineering (FSE) and NITRD/SPD Working Conference on the Future of Software Engineering Research (FoSER 2010). (2010)

9. Giese, H., Klein, F.: Systematic verification of multi-agent systems based on rigorous executable specifications. Int. J. Agent-Oriented Softw. Eng. **1** (2007) 28–62
10. Henkler, S., Hirsch, M., Priesterjahn, C., Schäfer, W.: Modeling and verifying dynamic communication structures based on graph transformations. In: GI Software Engineering. (2010)
11. Ivancic, F.: Modeling and Analysis of Hybrid Systems. PhD thesis, University of Pennsylvania (2003)
12. Henzinger, T.A.: Masaccio: A formal model for embedded components. In: IFIP International Conference on Theoretical Computer Science (TCS), LNCS1872, Springer, 549-563. (2000)
13. Burmester, S., Giese, H., Tichy, M.: Model-Driven Development of Reconfigurable Mechatronic Systems with Mechatronic UML. In Assmann, U., Rensink, A., Aksit, M., eds.: Model Driven Architecture: Foundations and Applications. LNCS, Springer Verlag (2005) 1–15
14. Métayer, D.L.: Software architecture styles as graph grammars. In: SIGSOFT '96: Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering, New York, NY, USA, ACM (1996) 15–23
15. Kramer, J., Magee, J., Sloman, M.: Configuring distributed systems. In: EW 5: Proceedings of the 5th workshop on ACM SIGOPS European workshop, New York, NY, USA, ACM (1992) 1–5
16. Allen, R., Douence, R., Garlan, D.: Specifying and analyzing dynamic software architectures. Lecture Notes in Computer Science **1382** (1998) 21–36
17. Kramer, J., Magee, J.: Analysing dynamic change in software architectures: A case study. In: CDS '98: Proceedings of the International Conference on Configurable Distributed Systems, Washington, DC, USA, IEEE Computer Society (1998) 91
18. Hungar, H.: Compositionality with strong assumptions, Mälardalen Real–Time Research Center (11 2011) 11–13
19. Baumgart, A., Böde, E., Büker, M., Damm, W., Ehmen, G., Gezgin, T., Henkler, S., Hungar, H., Josko, B., Oertel, M., Peikenkamp, T., Reinkemeier, P., Stierand, I., Weber, R.: Architecture modeling. Technical report (03 2011)
20. Meyer, B.: Applying "design by contract". Computer **25**(10) (1992) 40–51
21. CESAR SP2 Partners: Definition and exemplification of requirements specification language and requirements meta model. CESAR_D_SP2_R2.2_M2_v1.000.pdf on http://www.cesarproject.eu/fileadmin/user_upload/ (2010)
22. Damm, W., Hungar, H., Josko, B., Peikenkamp, T., Stierand, I.: Using contract-based component specifications for virtual integration testing and architecture design. In: Design, Aut. and Test in Europe (DATE 2011. 1–6
23. Bauer, J., Wilhelm, R.: Static analysis of dynamic communication systems by partner abstraction. In Nielson, H.R., File, G., eds.: Static Analysis, Int. Symposium, SAS 2007. Volume 4634 of LNCS., Springer (2007) 249–264
24. Winter, A., Kullbach, B., Riediger, V.: An overview of the gxl graph exchange language. In: Revised Lectures on Software Visualization, International Seminar, London, UK, UK, Springer-Verlag (2002) 324–336
25. Rensink, A.: The groove simulator: A tool for state space generation. In Pfaltz, J.L., Nagl, M., Böhlen, B., eds.: Applications of Graph Transformations with Industrial Relevance (AGTIVE). Volume 3062 of Lecture Notes in Computer Science., Berlin, Springer Verlag (2004) 479–485
26. Gezgin, T., Henkler, S., Stierand, I., Rettberg, A.: Impact analysis for timing requirements on real-time systems. In: Int. Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2014). (To be published)

Probabilistic Thinking to Support Early Evaluation of System Quality

Through Requirement Analysis

Mohammad Rajabalinejad*, Maarten G. Bonnema

Faculty of Engineering Technology,
University of Twente 2522 LW Enschede, Netherlands
*M.Rajabalinejad@utwente.nl

Abstract. This paper focuses on coping with system quality in the early phases of design where there is lack of knowledge about a system, its functions or its architect. The paper encourages knowledge based evaluation of system quality and promotes probabilistic thinking. It states that probabilistic thinking facilitates communication between a system designer and other design stakeholders or specialists. It accommodates tolerance and flexibility in sharing opinions and embraces uncertain information. This uncertain information, however, is to be processed and combined. This study offers a basic framework to collect, process and combine uncertain information based on the probability theory. Our purpose is to offer a graphical tool used by a system designer, systems engineer or system architect for collecting information under uncertainty. An example shows the application of this method through a case study.

Keywords: system; quality; uncertainty; design; evaluation

Nomenclature

μ	expected value
α	relative weight
d_i	a random number representing the system quality over the i-th requirement
d_{i_k}	a random number representing the system quality over the i-th requirement according to the k-th stakeholder
E	expected value
λ	relative weight of requirements
m	number of stakeholders
n	number of requirements
r_i	a random number representing the importance of the i-th requirement
r_{i_k}	a random number showing the opinion of k-th stakeholder over i-th requirement
s_k	a random number representing the importance of the k-th stakeholder
s_{k_j}	a random number showing the opinion of j-th stakeholder over k-th stakeholder
sq	a random number representing the system quality
Var	variance

1 Introduction

To deliver a quality system, a system designer should first identify, clarify, and document system requirements [1]. These tasks are performed in the earliest phase of a project life cycle and in the presence of a high level of uncertainty [2]. These requirements are not fixed and may change throughout the development stages [3]. On the other hand, some requirements e.g. maximization of benefits are explicitly or implicitly present in all design phases, and different terminology may be used for them. For example, design objectives or concept drivers are commonly used in the concept phase while the program of requirements or design criteria are more likely to be used in the embodiment phase. It is important to note that the requirements keep the focus of the design team on the most important design aspects or main needs and they provide references for the evaluation of system quality.

Therefore, system requirements have explicit roles through the design process. It is mainly because of the presence of a systematic approach [4] in this process and also because of the societal demands for meeting the standards [5] in engineered products. These are reflected in tools, processes and standards. An example is the popular method called the house of quality which relates user requirements to design requirements in order to ensure quality end-products. To achieve quality systems, designers need to define proper system requirements as early as possible [6] as they help judging the relevance of new information.

The evaluation of design alternatives are on the basis of these requirements. In other words, every design alternative has to be able to address the initial requirements. As a result, these requirements form criteria for evaluation of system quality. These design criteria may change through the design process and may have different degrees of importance. To assess system quality, a system designer has to rank them at the early stages of the design process. Ranking methods is of great value in decision models, and the use of multi criteria decision models (MCDM) typically involve criteria ranking.

1.1 Information elicitation

To define system requirements, identification of stakeholders is one of the earliest steps. A review research by Pacheco and Garcia [9] confirms that an incomplete set of stakeholders may lead to incomplete requirements. A system designer has to pay attention to the problems arising from the scope, understanding and validation of requirements [10, 11] in the course of communication with stakeholders.

Figure 1 presents the functional diagram for identification of stakeholders and communication with them. It shows some new stakeholders may be realized through the course of communication with already-known stakeholders. To document the stakeholder's needs and collected feedback , Salado and Nilchiani [12] suggest a set of questions for discovering new stakeholders in order to identify a complete set of stakeholders. Complex systems often include a relatively high number of stakeholders with different (conflicting) interests [13]. In such cases, the process of information

elicitation, documentation and integration is a necessity to achieve informative conclusions.

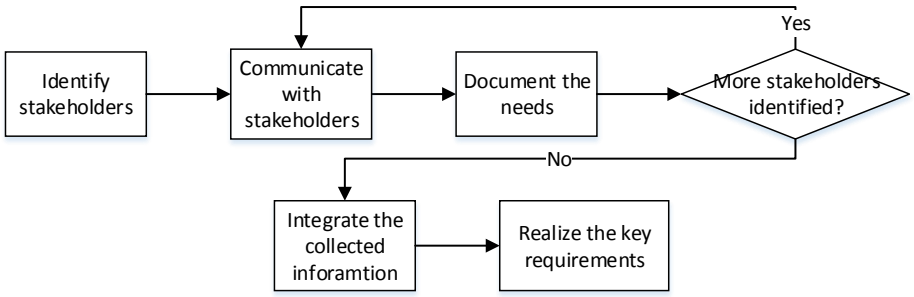


Fig. 1. The process of identification of stakeholders, communication with them, integration and prioritization of the collected data.

1.2 Ranking process

Ranking of requirements (or criteria) based on their importance is well discussed in decision models. The use of multi criteria decision models typically involves a systematic ranking process as for instance indicated in [4, 14]. The influence of the ranking process on final decisions is for example explained in [15]. A review of subjective ranking methods shows that different methods cannot guarantee accurate results. This inconsistency in judgment explains difficulty of assigning reliable and subjective weights to the requirements. A systematic approach for ranking is described in [16] that is a generalization of Saaty's pairwise structure [17]. Given the presence of subjectivity in the ranking process, sensitivity analysis of the design criteria is used to study the influence of variation and the ranking process on the decisions made [18]. Furthermore, some approaches e.g. the task-oriented weighing approach is effectively used. This approach is meant to limit the subjectivity of criteria weighting [19]. It suggests an algorithm to rank criteria objectively while considering the uncertainty in criteria weight [20]. The approach is based on introducing fuzzy numbers that imposes specified membership functions, which has been also used in [21, 22].

However, there is an obstacle for systems architectures or engineers in communication of the proposed methods with different stakeholders. The stakeholders can be individuals, corporations, organizations and authorities, with different fields/ levels of knowledge and experience [2]. The stakeholders have interest in the project and they desire to express their knowledge and expertise to improve the design. They also have expectations which have to be addressed at the end. Besides, it is advised to designers to rely on the experts in order to manage design uncertainties since it is proven that experts provide frameworks for making knowledge based decisions under uncertainty [23, 24]. This offers a human solution in terms of preferred alternatives. The uncertainty in importance of design requirements is also of human nature which should be reflected in the weighting process. To address these, we present the principles of our method through the next section.

1.3 Evaluation of system quality

To estimate the system quality, an intuitive method is used. Detail description of this method and its formulation emerge through the rest of this paper. It provides a consistent framework to value the system under uncertainty and observe how well the system addresses the stakeholder's needs. This outcome provide valuable sources for the system designer or system engineer to monitor the strong and weak point of the system. Figure 2 presents the functional flow for evaluation of system quality in a pluralistic approach where the stakeholders' opinion is fundamentally contributed to quality evaluation. In this perspective, communication plays an essential role and the proposed method aims to facilitate this communication.

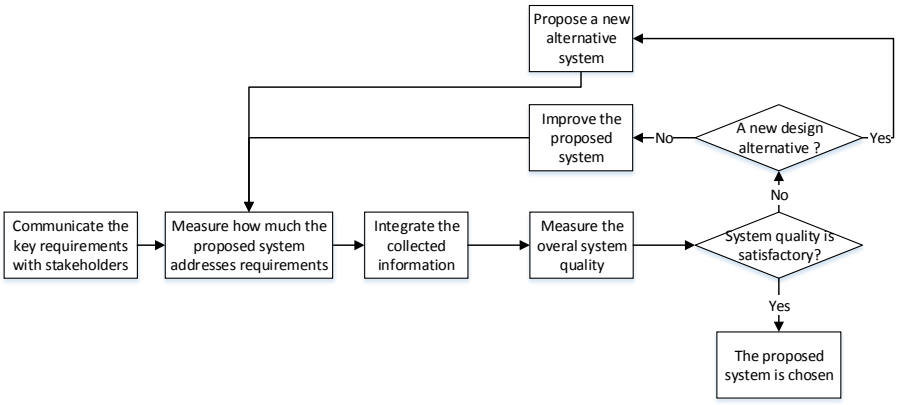


Fig. 2. The work flow for evaluation of system quality.

1.4 Presentation

We aim to present a realistic and intuitive approach that can communicate to people with different fields of knowledge and expertise. The method must be transparent, easy to implement and readily adaptable by different users. For this purpose, graphs are used to effectively communicate with different users. The format presented in Figure 3 identifies the importance of a requirement according to a stakeholder's opinion. The linguistic scale or the numeric scale can be used for the ease of communication, and one can assign the range of possible importance to a certain requirements.

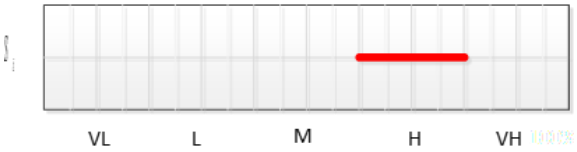


Fig. 3. An example of a stakeholder's opinion about the importance of the i-th stakeholder S_i .

A probability distribution function (PDF) is assigned to this recorded data. Symmetric opinions are assumed here in this paper as described in [25, 26] and the collected data is treated as a random variable with a Gaussian distribution aiming to achieve set of a stochastic weight factors.

2 Formulation

2.1 Ranking of Stakeholders

Having m stakeholders, each stakeholder evaluates the importance of all the stakeholders. This information is presented by stochastic variables $s_{k_1}, s_{k_2}, ..., s_{k_m}$, where s_{k_j} represents the opinion of j-th stakeholder over the importance of k-th stakeholder, and its expected value and variance are respectively shown by $\mathrm{E}[s_{k_j}]$ and $\mathrm{Var}[s_{k_j}]$. The expected relative weight and variation for the importance of each stakeholder is achieved by the following equations.

$$E[\alpha_k] = \frac{1}{\sum_{k=1}^{m} \mathrm{E}[s_k]} \sum_{j=1}^{m} \mathrm{E}[s_{k_j}] \tag{1}$$

$$\mathrm{Var}[\alpha_k] = \frac{1}{\left(\sum_{k=1}^{m} \mathrm{E}[s_k]\right)^2} \sum_{j=1}^{m} \mathrm{Var}[s_{k_j}] \tag{2}$$

2.2 Ranking of requirements

Now m stakeholders assess the importance of the i-th requirement r_i, and this information is represented by stochastic variables $r_{i_1}, r_{i_2}, ..., r_{i_m}$, where r_{i_k} presents the k-th stakeholder's opinion over the importance of the i-th requirement. As a result, the overall expected value and variation of the opinions over the importance of the i-th requirement r_{i_k} can be calculated by the following equations.

$$\mathrm{E}[r_i] = \frac{1}{\sum_{k=1}^{m} \mathrm{E}[s_k]} \sum_{k=1}^{m} \mathrm{E}[s_k]\,\mathrm{E}[r_{i_k}] \tag{3}$$

$$\mathrm{Var}[r_i] = \frac{\sum_{k=1}^{m} \mathrm{E}[s_k]^2\,\mathrm{Var}[r_{i_k}]}{\left(\sum_{k=1}^{m} \mathrm{E}[s_k]\right)^2} \tag{4}$$

2.3 Evaluation of system quality

A quality system must be able to address the initial requirements. Using the proposed method of this paper, the designer can quantify the stakeholders' opinion and evaluate how successfully the system addresses those requirements. For this purpose, stakeholders evaluate the system quality with regard to the system requirements and this information is labeled as $d_1, d_2, ..., d_i$, where d_i represents the stakeholders' opinion over the i-th requirement. The collected data is shown by stochastic variables $d_{i_1}, d_{i_2}, ..., d_{i_m}$, where d_{i_k} presents the k-th stakeholder's opinion over the importance of the i-th requirement. As a result, the overall expected value and variation of the opinions over the system quality with regard to the i-th requirement d_i is calculated by the following equations.

$$\mathrm{E}[d_i] = \frac{1}{\sum_{k=1}^{m} \mathrm{E}[r_k]} \sum_{k=1}^{m} \mathrm{E}[r_k]\,\mathrm{E}[d_{i_k}] \tag{5}$$

$$\mathrm{Var}[d_i] = \frac{\sum_{k=1}^{m} \mathrm{E}[r_k]^2\,\mathrm{Var}[d_{i_k}]}{\left(\sum_{k=1}^{m} \mathrm{E}[r_k]\right)^2} \tag{6}$$

And the overall system quality (sq) and its variation can be shown through the equations below.

$$\mathrm{E}[sq] = \frac{\sum_{i=1}^{n} \mathrm{E}[d_i]}{n} \tag{7}$$

$$\mathrm{Var}[\mathrm{sq}] = \frac{\sum_{i=1}^{n} \mathrm{Var}[d_i]}{n^2} \tag{8}$$

1. Algorithm

The block diagram of workflow for evaluation of system quality is shown by Figure 4. It shows three main steps to evaluate system quality. The first step, which is of essential importance, is to identify the stakeholders and their requirements. Then the stakeholders and the realized requirements are ranked. Having this data, the system quality is evaluated.

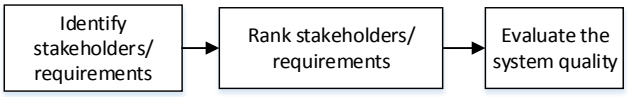


Fig. 4. The functional block diagram for the algorithm.

The following steps present the ranking process for requirements:

- List m stakeholders.
- Draw tables and list stakeholder $(s_1, s_2, ..., s_m)$ using the numeric or verbal format shown in Figure 3.
- Ask the stakeholders to fill the tables. This step concludes m series of tables. Use s_{k_j} format to label the collected information.
- Use Equation 1-2 and calculate $\mathrm{E}[\alpha_k]$ and $\mathrm{Var}[\alpha_k]$.
- List n requirements.
- Draw tables and list requirements $(r_1, r_2, ..., r_n)$ using the numeric or verbal format shown in Figure 3.
- Ask the stakeholders to fill the tables. This step concludes m series of tables. Use r_{i_k} format to label the collected information.
- Use Equations 3-4 to calculate $\mathrm{E}[r_i]$ and $\mathrm{Var}[r_i]$ for each requirement r_i.
- Draw tables and list requirements using the numeric or verbal format shown in Figure 3.
- Ask the stakeholders to evaluate the system against the requirements and fill the tables. This step concludes m series of tables. Use d_{i_k} format to label the collected information.
- Use Equations 5-6 to calculate $\mathrm{E}[d_i]$ and $\mathrm{Var}[d_i]$.
- Use Equations 7-8 to calculate the overall system quality and its variation.
- If new stakeholders or values are realized, reiterate from the first step. Reuse of the collected information is possible.

This process integrates the collected information and results an overview to a system designer for sorting the requirements based on the stakeholders' opinion. Next section presents an example application that shows the process and expected outcomes.

3 Example application

This section presents an example application to describe the proposed method. This example presents a stair-mobility project. This example shows an early estimation of the design quality in early phases of a project lifecycle where usually a high uncertainty level is present.

A company in cooperation with TUDelft defined this project, and a team of students worked on this project and an individual designer finalized it. The aim of this project was developing a concept for chair stair-lifts used by adults in the Western Europe with minor disabilities. This could represent a target group that start feeling pain in hips, knees or ankles but also consider fatigue and fear issues during the ascend or descend of staircase.

Based on the stakeholder's requirements and designers' vision, several requirements were defined to ensure desired functions. For demonstration purpose, we refer to two of them: natural interaction and ergonomics. Natural Interaction prevents stig-

mata surrounding stair lifts, and ergonomics ensures that the product generates a natural interaction with its user. These requirements are illustrated in Figure 5. This figure shows the opinion of three stakeholders, and they quantified the stair lift system using our proposed method. Here in this paper they are evenly graded for demonstration purpose, and a numerical scale has been used in the figure.

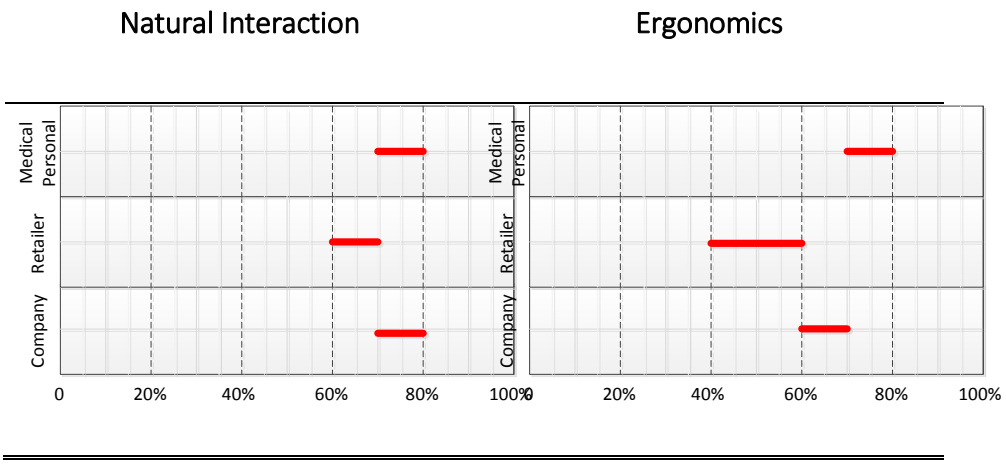


Figure 5. (a) Three stakeholders present their expert opinion on the proposed design for stair-lift system.

Applying the algorithm explained in Section 2 results in Table 1. This table presents the integrated and concluding results. Two design requirements and three expert opinions on these requirements are presented in this table.

Table 1. This table presents the integrated results for requirements and system quality.

System requirements			System quality	
Requirements	Expected values for requirements (E[r_i])%	Standard deviation of requirements (Var[r_i])%	Expected value of quality (E[sq])%	Standard deviation of quality (Var[sq])%
Natural interaction	71.6	17.3		
Ergonomics	63.7	24.5		
			67.7	29.9

Conclusions

This study describes a methodology to measure system quality on a pluralistic basis. It embeds the importance of design stakeholders and requirements. The proposed method enables and encourages a designer to communicate with stakeholders or experts and collect certain or uncertain information, combine this information and valuate system quality. The application of this method has been shown through the ColdFacts project.

The proposed approach promotes the probabilistic thinking and establishes the principals of a method for using uncertain information based on the probability theory. This method facilitates information collection and information integration in large, complex or high-tech systems[13]. Furthermore, it can be integrated with some currently used methods in system design or systems engineering.

References

1. Haskins, C. *Systems engineering handbook*. 2006. INCOSE.
2. Rajabalinejad, M. and C. Spitas, *Incorporating Uncertainty into the Design Management Process.* Design Management Journal, 2012. **6**(1): p. 52-67.
3. Ronkainen, I.A., *Criteria changes across product development stages.* Industrial Marketing Management, 1985. **14**(3): p. 171-178.
4. Pahl, G., W. Beitz, and K. Wallace, *Engineering design: a systematic approach*. 1996: Springer Verlag.
5. Sen, P. and J.B. Yang, *Multiple criteria decision support in engineering design*. Vol. 4. 1998: Springer London.
6. Spitas, C., *Analysis of systematic engineering design paradigms in industrial practice: A survey.* Journal of Engineering Design, 2011. **22**(6): p. 427-445.
7. Balachandra, R. and J.H. Friar, *Factors for success in R&D projects and new product innovation: a contextual framework.* Engineering Management, IEEE Transactions on, 1997. **44**(3): p. 276-287.
8. Roozenburg, N.F.M. and J. Eekels, *Product design: fundamentals and methods*. 1995: John Wiley & Sons.
9. Pacheco, C. and I. Garcia, *A systematic literature review of stakeholder identification methods in requirements elicitation.* Journal of Systems and Software, 2012. **85**(9): p. 2171-2181.
10. Christel, M.G. and K.C. Kang, *Issues in requirements elicitation*. 1992, DTIC Document.
11. Heemels, W., et al., *The key driver method.* Boderc: Model-Based Design of High-Tech Systems, edited by W. Heemels and GJ Muller, 2006: p. 27-42.
12. Salado, A. and R. Nilchiani, *Contextual-and Behavioral-Centric Stakeholder Identification.* Procedia Computer Science, 2013. **16**: p. 908-917.
13. Heemels, W., E. vd Waal, and G. Muller, *A multi-disciplinary and model-based design methodology for high-tech systems.* Proceedings of CSER, 2006.

14. Whitten, J.L., V.M. Barlow, and L. Bentley, *Systems analysis and design methods*. 1997: McGraw-Hill Professional.
15. Barron, F.H. and B.E. Barrett, *Decision quality using ranked attribute weights.* Management Science, 1996. **42**(11): p. 1515-1523.
16. Takeda, E., K.O. Cogger, and P.L. Yu, *Estimating criterion weights using eigenvectors: A comparative study*. European Journal of Operational Research, 1987. **29**(3): p. 360-369.
17. Saaty, T.L. and L.G. Vargas, *The logic of priorities: applications in business, energy, health, and transportation*. 1982: Kluwer-Nijhoff.
18. Barzilai, J., *Deriving weights from pairwise comparison matrices.* Journal of the Operational Research Society, 1997. **48**(12): p. 1226-1232.
19. Yeh, C.-H., et al., *Task oriented weighting in multi-criteria analysis.* European Journal of Operational Research, 1999. **119**(1): p. 130-146.
20. Buckley, J.J., *Ranking alternatives using fuzzy numbers.* Fuzzy Sets and Systems, 1985. **15**(1): p. 21-31.
21. Tsai, W.C., *A Fuzzy Ranking Approach to Performance eEaluation of Quality*. 2011. Vol. 18. 2011.
22. Mitchell, H.B., *Rnking-Intuitionistic Fuzzy Numbers.* International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2004. **12**(03): p. 377-386.
23. Zimmermann, H.J., *Fuzzy sets, decision making and expert systems*. Vol. 10. 1987: Springer.
24. Rajabalinejad, M. *Modelling dependencies and couplings in the design space of meshing gear sets*. 2012.
25. Choy, S.L., R. O'Leary, and K. Mengersen, *Elicitation by design in ecology: using expert opinion to inform priors for Bayesian statistical models.* Ecology, 2009. **90**(1): p. 265-277.
26. O'Hagan, A., J. Forster, and M.G. Kendall, *Bayesian inference*. 2004: Arnold London.

	Concept 1			
	Interaction (*w=0.6*)		Ergonomic (*w=0.4*)	
	Expected value (%)	Limits (%)	Expected value (%)	Limits (%)
Company	75	80-70	65	70-60
Retailer	65	70-60	50	60-40
Med. Personal	75	80-70	76	80-70

System Engineering on 3DEXPERIENCE Platform - UAS Use Case

Frédéric CHAUVIN, Gauthier FANMUY

Dassault Systèmes
10 Rue Marcel Dassault, 78140 Vélizy-Villacoublay

System Engineering has become increasingly complex over years. Not only the designed product itself must fulfill more requirements (functional, performances, ergonomics, interoperability, safety, …) but additional constraints such as environmental friendliness, globalization, OEM / Suppliers relationships, intellectual property, regulatory compliance and many more must be integrated in the design process. This implies an extended multidisciplinary collaboration between system engineers and beyond with all participants and stakeholder of the project.

The Unmanned Aerial System (UAS) use case illustrates how a modern tool suite can dramatically optimize the product development process. All the required information is made accessible in an integrated environment to ensure collaboration, project management, wise decision making and ultimately to help to imagine sustainable innovations capable of harmonizing product, nature and life.

1 Introduction

Civilian UAS is fast growing market which promises countless usages such as aerial imagery (marketing, advertising, journalism,…), photogrammetry (GIS), infrastructure inspection, crop science, fire fighting, search and rescue…
Each scenario has a specific mission profile and associated set of constraints.

	Scenario 1	Scenario 2
Purpose	Sport event coverage	Railway inspection
Flight Plan	On demand (Manual control)	predefined waypoints
Flight Distance	<100m	Beyond line of sight
Typical cruise speed	< 40 km/h	> 60km/h

Take Off	Vertical	Hand launched or catapult (mandatory above ~3kg)
Landing	Vertical	Belly landing or parachute
Payload	1.5Kg (DSLR camera)	300g (compact camera)
Camera Gimbal	3 axes (mandatory)	1 or 2 axis (optional)
Required endurance	20mn with payload (repeatable with quick battery replacement)	> 1h
Safety risks	Very high (Flight near bystanders)	Medium (Flight in open country)

With today's solutions, scenario 1 is typically addressed with a multicopter such as DJI S1000 (Fig. 1) while scenario 2 is addressed with a fixed wing aircraft such as Delair-Tech DT-18 (Fig. 2).

Fig. 1 DJI S1000

Fig. 2 Delair-Tech DT-18

Multicopter are easy to operate as they can take-off and land vertically and provide augmented stability. They can also hover which make them very suitable for steady aerial shooting. But they have limited endurance and cruise speed.

On the other end, fixed wing aircrafts are aerodynamically efficient. They have better cruise speed and endurance. But they are more difficult to operate due to take-off and landing constraints.

These very specific and radically different architectures prevent flexible usage and increase cost of ownership as operator is forced to own multiple UAVs to support various scenarios.

A new generation of Vertical Take Off and Landing (VTOL) UAS is considered to bring the benefits of fixed wing aircraft to the flexibility of multicopter usage. This innovation requires a robust system engineering methodology and tool suite which is explained and illustrated in this article.

2 System Engineering on the 3DEXPERIENCE Platform

2.1 The Systems Engineering promise: early decision makings to build cost-effective solutions

One of the main challenges is to build a cost-effective solution that meets stakeholders' needs and constraints. The Systems Engineering promise is to enable trades-off for early decision-makings that select from various requirements and alternative solutions on the basis of net benefit to the stakeholders.

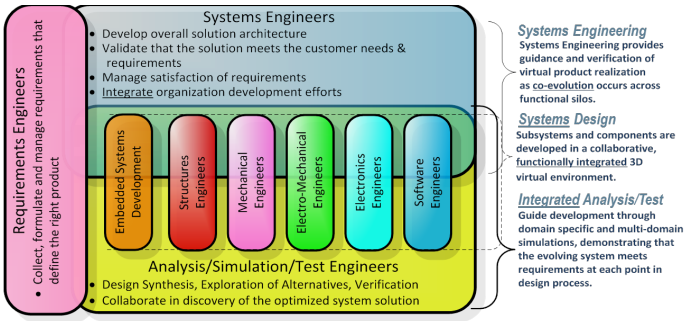


Figure 3 - Systems Engineering guides a collaborative team working in a shared knowledge space for virtual system co-evolution

To reach this goal is in practice complex as several inhibitors may influence:

- The maturity of the organization in Systems Engineering may vary from one department to another, for one team to another.
- The Engineering is in silos: collaboration between teams is weak and is mainly based on a "document" or "deliverable" basis.
- The tool suite that support the Systems Engineering process have no or low coupling

In such contexts, trades-offs are often based on informal or incomplete criteria and rely on the knowledge of a set of architect experts.

2.2 RFLP

The acronym "RFLP" (Requirements, Functional, Logical and Physical view) has been since the 80s a known description of the core elements of Systems Engineering. It has been supported in by MIL-STD 499B "Military Standard –

Systems Engineering" (first version 1974, draft revision 1994) that was then replaced by IEEE/1220 (first version: 1994, revised in 1999 and 2005) "Standard for Application and Management of the Systems Engineering Process".

RFLP provides a unique data referential to support a Systems Engineering process with requirements (R), functions (F) and logical product definition (L) such as components including 0-1D models, physical product definition (P) including CAE multidimensional models.

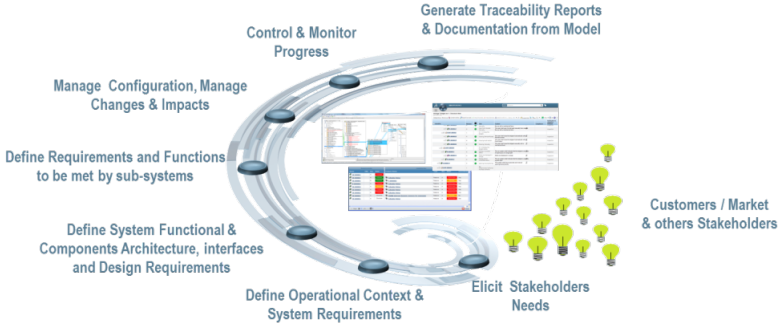


Figure 4 - Typical Systems Engineering Process

RFLP is a framework that supports a Model Based Systems Engineering process: it is a unified system definition with 4 fundamentals facets:

- Requirements: describes all the requirements that a system has to fulfill, from stakeholders' requirements to system & design requirements,
- Functional: describes the system services, and the functional architecture with functions and flows that the components of the system must provide,
- Logical: describes the components architecture with components of the system, theirs interfaces and the allocated functions & flows.
- Physical: defines the life-like system components, including the disciplines 3D Modeling (Mechanical, Electrical, Fluidics…)

The RFLP implement link model provides a full traceability from Requirements to Physical in both directions.

2.3 The 3DExperience Platform for Systems Engineering

The 3DEXPERIENCE Platform is a business experience platform available on premise and in public or private cloud, and whose purpose is to enable 3DS' customers to create experiences for their ultimate customers or consumers. It is a new generation of platform that enables all disciplines in a company and its ecosystem.

The 3DEXPERIENCE Platform is built upon the applications from different brands that are connected through a common database backplane:

- Social and Collaborative applications.
- Model Based Systems Engineering and 3D Modeling applications Content & Simulation Applications
- Information Intelligence Applications

The 3DEXPERIENCE Platform for Systems Engineering consists of the main following integrated applications:

- Traceable Requirement Management to develop and manage requirements and tests with traceability all along the system life cycle
- Functional & Logical design to define functional and components architectures with interfaces definition, in a model based approach
- Behavior modeling to add dynamic and/or static models from various engineering fields (electrical, fluids, mechanical…) to the components. These models can be based on the open source language Modelica (1) designed directly in the 3D Experience Platform or models coming from external sources thanks to the FMI exchange standard (Function Mock up Interface)
- Virtual physical prototypes to run experiences including 3D, HIL, SIL codes into in V&V process
- Report Generation to generate documentation from the objects managed in the 3DEXPERIENCE Platform

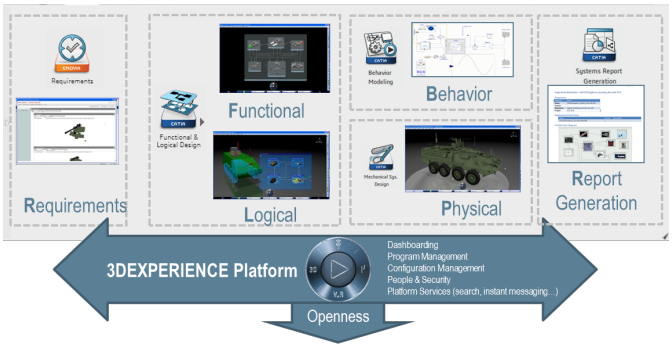


Figure 7 - The 3DEXPERIENCE Platform for Systems Engineering

3 UAS Use Case

This UAS use case has been developed mainly by a single engineer who had initially none to limited skills on UAS technologies, RFLP and Modelica language.

This represents an effort of:-

- 2 months on training mainly on RFLP and Modelica based behavior modeling Apps.
- 2 months on UAS documentary research, mainly on autopilot, aerodynamics, propellers, brushless motors and controllers, MAVLink Ground Control Station communication protocol.
- 8 months of dynamic behavior and 3D modeling on various tradeoffs.

Additional engineers have contributed a cumulated effort of 3 months, mainly on FMU (2) development (UDP communication), 3D Styling and 3D virtual environment creation.

3.1 Introduction

The original idea is to combine a multicopter architecture to provide VTOL capability with a flying wing to provide aerodynamics efficiency and endurance.

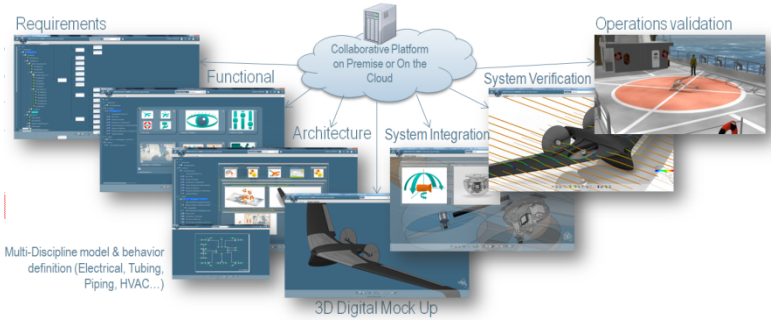
The key drivers considered for the UAS alternative architectures are:

- Aerodynamics efficiency mainly driven by the 3D exterior shape and center of gravity position relatively to aerodynamic center.
    - The key metric is the “Lift to Drag” ratio which characterizes the efficiency of the wing to produce the lift required to sustain the aircraft versus the drag which has to be compensated by the propulsion.
- Stability. It is mainly driven by the airfoil choice and the static margin, i.e. the distance between aerodynamic center and center of gravity (CoG) divided by the mean aerodynamic chord. The usual process is to adapt the CoG by placing massive mechanical parts and equipments at the appropriate location. But on this design, as the CoG is constrainted to be near the center of thrust of the multicopter, this implies strong constraint on the aircraft shape to make sure that the aerodynamic center will be located rear the CoG.
- Propulsion
    - engine layout : Y3, Y6, X4, +4, X6, X8

  - Propeller size and pitch
  - Maximum continuous power
  - Thrust to weight ratio. It is generally admitted that is must range from 1.5 to 2 for provide safe multicopter handling.
- Use of a maximum of components on the shelf (COTS) to reduce costs.

3.2 RFLP

Within 3DEXPERIENCE integrated environment, the RFLP referential is instantiated to support the end to end design process.



3.3 Models

Multi dimensional (0..3D) models are developed in parallel to provide the required information to evaluate most critical key performance indicators (KPIs) on:

- Aerodynamics efficiency
- Thrust to weight ratio
- Mechanical Design feasibility
- Equipments and payload space allocation
- Mission capabilities
- Total cost of ownership

3.3.1 3D Design

A set of preliminary 3D design templates is created to provide external shape to CFD analysis and study propulsion layout, equipments and payload space allocation and center of gravity. Template instantiation and parameterization allows rapid iterations and alternatives studies (Fig 8).

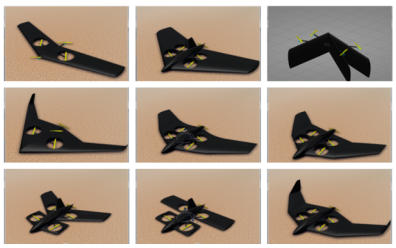


Figure 8 - rapid 3D ideas sketching based on design templates

3.3.2 Multi-body dynamics

Thanks to 3DEXPERIENCE integration, 3D Digital Mockup can be translated into Modelica model, including body mass and inertia and kinematics joints definition. Resulting model can be completed with external forces such as aerodynamics effects and propulsion.

3.3.3 Propellers

The model uses the advance ratio J equation (3) and two two dimensional tables of experimental data (4) to relate J, rotational speed N, torque and thrust (Fig. 9).

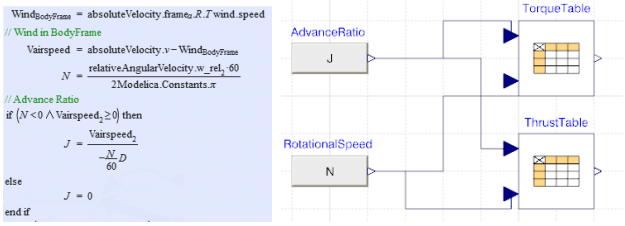


Figure 9 – Propeller model based on experimental data

3.3.4 Electric Motors & Batteries

The SmartElectricDrive library is used with 3 levels of accuracy (quasi stationary, transient and full BLDC). These 3 models are attached to the same logical system component. The model used for simulation is chosen at runtime depending on expected accuracy and performance needs.

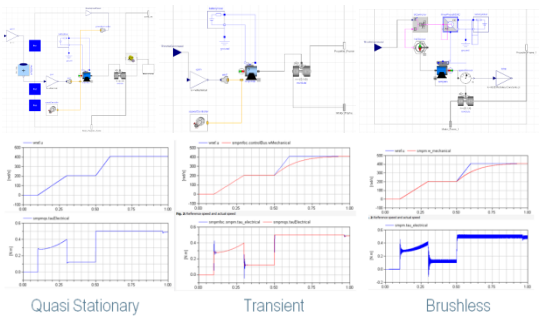


Figure 10 – multi level Modelica models and their respective dynamic response

3.3.5 Flight Dynamics

A Modelica model implements the 6 equations (5) (Fig. 11) to compute aerodynamics forces and moments to be applied on UAV body. These equations use a set of aerodynamic coefficients so called "polar" such as "lift vs. drag" or "pitch moment vs. alpha" that must be computed by dedicated CFD software.

$$Lift = \frac{1}{2}\rho S V^2 (Cz(\alpha) + Cz_{flap} + Cz_{ElevatorDeflection} + \cdots)$$

Fig. 11– Lift equation (partially developed) derived from Bernoulli equation

3.3.6 Flight Management System

Multicopter stabilization is known to be sensitive to inertial measurement unit accuracy and delay (gyroscope and accelerometers sensors typical refresh rate is 200 to 400HZ) and to motor ESCs refresh rate (50 to 400HZ or more). It is therefore critical to properly model the frequencies of the multiple stages of control loop in addition to sensor noise, quantization and bias.
The hybrid discrete/continuous solving capability of Modelica is used thanks to the LinearSystem2 library which provides PIDs, noise, filters models which can be easily switched from continuous to discrete on a global or per object basis. Modelica technology takes care of properly combining discrete models (control) with continuous physics models.

The multicopter control laws are superimposed with fixed wing aircraft control laws (Fig 12).

Hover to forward flight transition sequence:

1. rear engines are tilted in an fixed duration arbitrary chosen to leave the aircraft enough time to accelerate to minimum flight speed
2. When aircraft exceeds stall speed, forward engines are stopped
3. Then engines doors are closed.

Forward flight to hover transition sequence:

1. rear engines are tilted to horizontal position
2. When aircraft approaches stall speed, forward engine doors are opened.
3. Once the doors are opened, the forward engines are started.

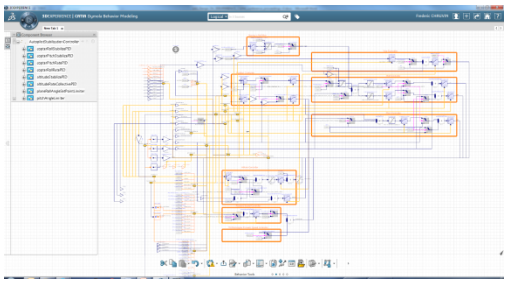


Fig 12.Overview of hybrid multicopter – fixed wing stabilization control laws

Two additional control laws have been developed:

1. Improved yaw control. On a traditional multicopter, yaw control is achieved by creating torque about yaw axis by increasing the rotational speed of engines which rotate in opposite direction of expected yaw rotation and simultaneously decreasing opposite engines to keep the vehicle at constant altitude. But on large multicopter, the propeller torque might be insufficient and result in sluggish control and responsiveness. As the two rear engines of the proposed VTOL design can be individually tilted, yaw responsiveness is improved by slightly tilting the engine in opposite direction (Fig 13). CATIA System simulations have demonstrated that a ⁀10° tilting range allows doubling the yaw rate and yet improving the yaw start/stop responsiveness.

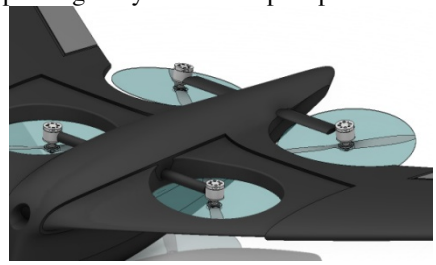


Fig 13. Improved yaw control thanks to opposite engine tilting.

2. Wind handling during hover. Wind during hover will tend to push the vehicle in opposite direction. To maintain position, traditional multicopters autopilot will incline the vehicle to create lateral force in opposite direction from

wind. The wing positioned with low or negative alpha angle will produce negative lift. Additional power must be provided to conpensate this negative lift and this reduces endurance. To avoid this drawback, multicopter pitch control law is modified to constrain pitch angle to a fixed positive alpha angle. This makes the assumption that the UAV is always headed into the wind thanks to manual or automated control. The resulting drag force is compensated by slightly tilting the rear engines (Fig 14). Forward/rear movement is controlled by adjusting the tilt angle. CATIA System simulation has demonstrated that a 10m/s wind with a constrained $\mapsto$=4.5° produces 1/3 of the thrust required to lift the aircraft to the benefit of endurance.

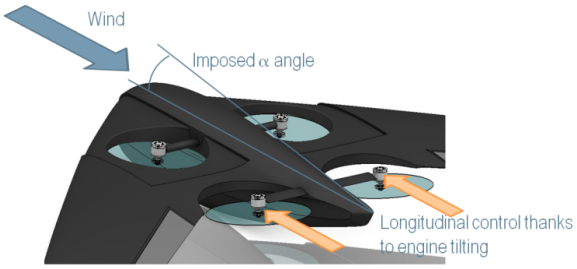


Fig 14. Longitudinal control and use of wind to lift aircraft during hover.

3.4 Computational Fluid Dynamics (CFD)

SIMULIA Fluid Scenario App is used to explore flight domain at various speed, angle of attack, sideslip angle, and engine doors position. This allows computing surrogate models (polars) used in Modelica flight dynamics model (Fig 14).

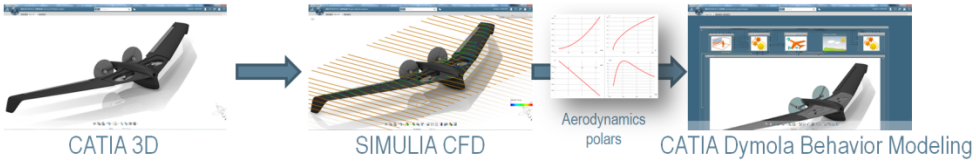


Figure 14 – 3D to CFD to System simulation inside 3DEXPERIENCE

3.5 Ground Control Station (GCS) link

A GCS (6) is a software operated from the ground to remotely control the UAV. It allows monitoring the UAV system parameters and health status, defining the mission and sending the mission to the UAV. MAVLink (7) is an UDP-based protocol specifically designed to handle communication between the GCS and the UAV (Fig 15). A generic FMU model is developed to provide UDP networking. It

is then customized to describe the messages with an xml file. This allows extending the communication (message content, periodicity or event based send/receive) with no need to recompile the FMU code.

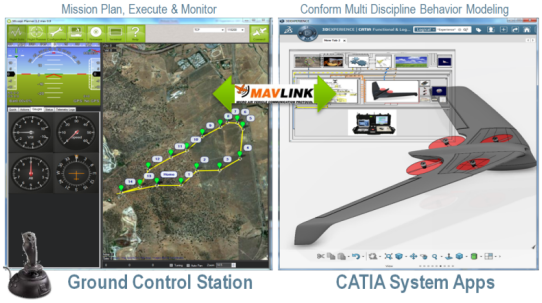


Fig 15. Ground Control Station and UAS communication

3.6 3D Virtual Environment

CATIA is connected to 3DVIA (Fig 16) via a FMU using a TCP sockets. CATIA periodically sends the position and attitude of the UAV and additional information such as elevons position, engines speed to animate the UAV in 3DVIA scene.

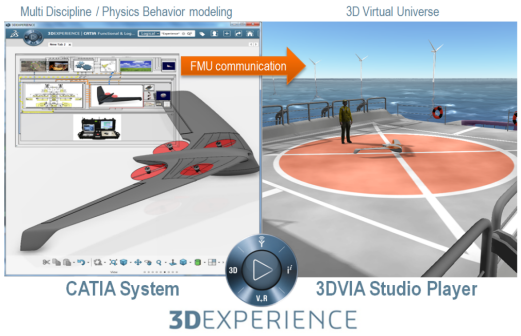
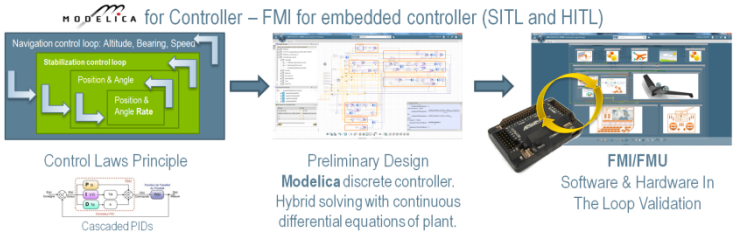


Fig. 16

3.8 Autopilot SITL/HITL

The flight control laws are modeled (discrete and/or continuous mode) using modelica. Once validated, it is convenient to replace this preliminary design by the actual C code running on PC (SITL) or running on real hardware (HITL). This is implemented with an FMU connecting the controller to the CATIA plant through UDP sockets.



4 Conclusion

The complete UAS mission "play" experience (Fig 17) is simulated in real time on a single laptop (UAV virtual model in CATIA System, real Ground Control Station software and 3DVIA 3D virtual environment).

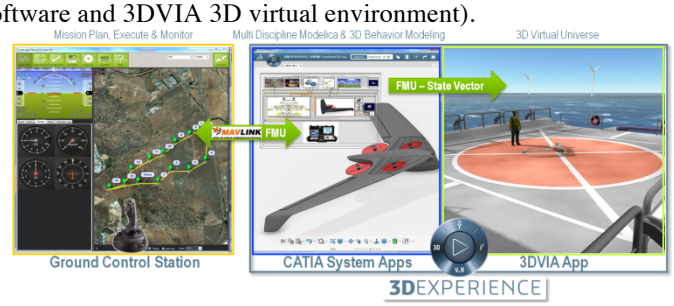


Fig. 17 UAS 3DEXPERIENCE

Further work will include optimization thanks to SIMULIA Process Composer (previously known as iSight) to systematically explore the UAS alternatives and fine tune their design KPIs.

3DPEXPERIENCE provides a holistic RFLP-based framework to support end to end system engineering processes in complex multi-discipline organizations.
UAS use case demonstrates that this tool suite is not restricted to largest companies and can be efficiently applied to small teams, especially on the Cloud.
Ease of use and fluidity of the user experience, best in class modeling capabilities unleashes creativity to surpass the status quo. 3DEXPERIENCE provides a unique collaborative business platform and defines the new standard for product innovation.

References

1. **Modelica**® is a non-proprietary, object-oriented, equation based language to conveniently model complex physical systems. www.modelica.org

2. Functional Mock-up Interface / Functional Mock-up Unit. https://www.fmi-standard.org/

3. Advance_ratio http://en.wikipedia.org/wiki/Advance_ratio

4. APC Propellers. http://www.apcprop.com/v/downloads/PERFILES_WEB/datalist.asp

5. Joseph M. Cooke, Michael J. Zyda, David R. Pratt, Robert B. McGhee. NPSNET: FLIGHT SIMULATION DYNAMIC MODELING USING QUATERNIONS. Int. J. Digit. Libr. 1 http://www.movesinstitute.org/~zyda/pubs/Presence.1.4.pdf (1994)

6. QGroundControl http://qgroundcontrol.org/

7. MAVLink, a very lightweight, header-only message marshalling library for micro air vehicles. http://qgroundcontrol.org/mavlink/start

Engineering a Parent-System, Designed to Generate Complex Sub-Systems in the Field of Defense Planning[1]

Wolfgang Peischel

National Defence Academy/Austrian Military Journal, 1070 Vienna, Stiftgasse 2a, Austria
wolfgang.peischel@bmlvs.gv.at

Abstract

The main message and added value of this analysis lies in the development of a possible structure of defense planning, plus process logics, and in providing inherent functional principles of a "system-generating system" in the field of defense planning in a broader sense. This includes the identification of the interrelations of the researched basic principles.

The present research starts with the hypothesis that system development, particularly in highly aggregated social entities, is usually initiated by organizational structures that represent a complex system in themselves. This paper attempts to address the inherent functional logics of this specific "system of systems", which is the precondition for a successful development and for any control of systems and system-modeling per se.

The described system focuses on creation, control and further development of those sub-systems that provide an adequate reaction to existential threats and future challenges in unpredictable, uncertain situations. Functional principles of military system-planning will be deduced, analyzed, and presented in an abstraction that still allows for practical application in the private decision-making sector, as well. A possible civilian benefit might be gained, where these sets of skills (a specific military "unique selling proposition") are in demand.

Military system planning is based on specific functional principles that are tailored to leadership-decisions and system control in threatening, time-critical, and unforeseeable situations, usually in a volatile environment.

Attempting to explain according to which military scientific deductions a "system-generating system" in the area of defense planning could be developed, it will be shown in which areas military/leadership-science can offer research results also to civilian system development and where defense planning could benefit from other scientific branches.

Into the direction of private economy an insight is to be given, according to which system-logic military decisions are made respectively which basic principles guide planning-/ defense procurement-processes.

1 Good reasons to maintain independent, uniquely "military" development logics

Military system development is exposed to a number of unfavorable influences of all three types of institutional isomorphic organizational change.[2]

The requirement of measurability of public administration performance, e.g. the pressure to achieve maximum cost-efficiency by outsourcing and ongoing organizational changes that are often triggered by political pressure, leads to a situation that Di Maggio/Powell describe as "coercive isomorphism".

The unclear perception of a hardly predictable, future European threat situation, seduces national armed forces to "mimetic processes" that aim at unreflected copying and adapting of role profiles, regardless of thorough strategic evaluation, and regardless of whether the capabilities needed to cope with *certain* geopolitical challenges are required in the *current* situation of a respective state.

The tendency towards professionalization of military organizational structures forces higher military leaders to align themselves with the skills of managers in private businesses, whereby, according to Di Maggio/Powell, the normative component of isomorphism is addressed.

However, armed forces, in sharp contrast to the private sector, have to follow the primacy of politics, but at the same time will have to preserve their specific, unique and independent assessment and system building logic, to develop it further, based on military requirements and to maintain their own code of values that necessarily has to be different from the one used in civilian society.

Stephan De Spiegeleire states that defense planning reality shows a lack of human and social science components.[3] When planning systems are responsible for the definition of long-term strategic goals, mirroring a level of systemic creativity, which allows to gain a distinct advantage over the (assumed) adversary, human-scientific and in particular philosophy-related approaches will be needed in addition to those that natural sciences provide.

This paper offers a model[4] by which a military-specific, independent "system generating system" in the field of defense planning can be designed, that builds on a military/leadership-scientific basis.

2 An abstract, ideal-typical structure of defense-planning and its inherent functional logic

Current literature pays too little attention to the fact that system-creation as well as system management, is carried out by an organizational structure, which in itself represents a complex system. It also is neglected that inappropriate (organizational, methodical, hierarchical) alignment of the generating system is often the reason why the attached and subordinated systems do not work properly. A first result should explain, (a) how the core functional principles of the generating system in the area of defense planning work, (b) what problems can be identified, and (c) which of these solutions are usable for any civilian application.

Regardless of whether it is in the field of political leadership, diplomacy, private economy or public administration, planners complain about a missing strategic long-term orientation.

This, among other reasons, might be caused by the fact that the future becomes harder to predict, environmental conditions turn out more volatile, competition gets tougher and resources scarcer. And there are the early errors, already buried in the initial development cycles, which have sped up in a way that errors, coming from false or too short-sighted strategic goals, start to take revenge later on, punishing those who were responsible for faulty analyses and definitions in the first place (yet within their term of responsibility). In addition, cycles change, so do requirements, so do demands.

Particularly at a time of international networking, multipolarity, globalization and asymmetric threats, the value of any generating system lies primarily in the farsightedness of its strategic objectives and the level of creativity it produces. Throughout the whole process, a high level of creativity is indispensable in order to develop the proposed or existing system further, so that it achieves superiority over competitors. Creativity is also needed to find strategic objectives, which preferably permit a non-confrontational goal achievement ("Blue Oceans"), and force competitors to react.

Thus, two principle options of system development can be identified – namely a gradual further development (adaptation), based on the already existing functional principles, or a fundamentally new idea, resulting in a new generating system, or at least in contributing advanced components to an already existing system.

Especially in military planning it is absolutely vital to distinguish reliably and at an early stage for how long it makes sense to adapt and adjust (modify) an existing system or principle (doctrine) and from when on it has to be changed over to a fundamentally new one. This does not only hold true for technological developments but also for other security-political paradigms, operational concepts, command/force structures and planning processes per se. The "Clipper-phenomenon" shows, how in the mid-19th century, the development of lighter and faster cargo-sailing-ships was hailed as a great technological achievement and widely praised as a success of the system "sail-ship". However, in its final phase, it (the clipper) became a last swash of a passed technology, a final harbinger of the end of a line which would delay the steam ship only by a few years.

Generating systems, in the field of defense planning, need to train leaders in the awareness and recognition of such developments. Regarding the identification of new developments (and possible new threats) the observance of trends is mandatory; here, civil sciences can provide the necessary "early warnings".

Since any further development of systems adheres to the same judgment processes, the same mathematical, rational and physical rules as well as resource conditions for all competitors, superiority will only be achieved when a winning-margin in creativity can be gained or when one opponent makes a mistake. But such a grave error - as military history and leadership doctrine show - should not be taken for granted by system planners.

This deduction shows, that creativity is a guiding principle to which a generating system, not only in the field of defense planning, should be aligned, in order to

achieve planning- and leadership-superiority, or as Henry Ford had explained in simpler terms, “If I had asked people what they wanted, they would have said faster horses”.

Fig. 1. Ideal-Typical Organizational Structure of a Fictitious MoD

Regarding the concrete search for new principles which are to provide a defense planning system with planning, command and control superiority, one can start from a typical system-architecture of a fictitious European Ministry of Defense (MoD). This starting point might not represent an actual organizational structure of a real national MoD, but should be understood as an ideal-typical structure, derived from a military-scientific leadership doctrine that responds to analyzed requirements for planning systems. Naturally it would mirror typical structures of existing defense ministries. (See figure 1)

The main advantage of this system-architecture lies in the fact that it supports a process-oriented defense planning approach. The organizational elements of defense planning are congruent with the classical planning process steps, following the typical sections like concept planning, definition of military strategic goals, research and development and capability/force-planning. These organizational elements are matrix-type structured and cross sectionally responsible for all services and branches of services. They are, with regard to defense planning and achievement, subordinated to the top defense planner, who is solely responsible for the entire defense planning process. This principle is also mirrored in the process-oriented structure of the European Defense Agency (EDA), and it meets the general trend of new partition-schemes within the capability planning methods that De Spiegeleire analyzes[5].

The organizational elements of defense planning, defense procurement and operations are directly subordinated to the Chief of the Defense Staff (CHOD). This ensures a dialectic interaction between planning and procurement, whereby a mutual

blockade between the latter can be successfully avoided. Planning, procurement and operations represent steps of a process that directly feeds back requirements and lessons learned from earlier programs and from current operations into the capability planning directorate. The logical linkage of both principles represents the ideal rationale for combining defense planning in a narrower sense, defense procurement and operations, to defense planning in a broader sense. Above that is the logical process chain "capabilities procurement operations", indispensable for a reliable and realistic life-cycle management (cf. De Spiegeleire, Trend 2. Towards Life-cycle Capability Management).

One of the core principles for the alignment of generating systems in the field of defense planning lies with the deliberate separation of creative steps and their follow-on implementation. In the suggested model, this principle applies fully to the relations of (a) defense planning and procurement, and (b) fundamental planning and capability planning. The organizational element, responsible for the development of a military strategy, contributes to the defense planning-relevant policy. By doing so, it shortens the distance between political leadership and military planners via the Chief of the Defense Staff.

The capability/force planning-element combines and harmonizes the personnel, materiel, infrastructure and educational aspects of deduced capabilities and thereby avoids incorrect planning, usually caused by an overestimation of the materiel component (cf. De Spiegeleire, Trend 1. Towards a Broader Definition of 'Capability').

The broader scientific basis on which modern defense planning is built, is maintained by including the organizational element - research and development (R&D) - into defense planning. The R&D element is responsible for the corresponding process step and, in close cooperation with the research and educational level (like defense universities, industry, think tanks, etc.), provides the scientific basis for the armed forces and has an impact on military/leadership science, the relevant humanities and social sciences as well as natural sciences (cf. De Spiegeleire, Trend 5. Broadening the Scientific Base).

3 Generating a parent-defense planning system and derived sub-systems

An ideal-typical system architecture of a fictitious European MoD has been offered above. Now it must be decided which elements serve the generating system of defense planning and which ones primarily support its generated sub-systems (see figure 2). In order to make the entire network between the generating parent-system and the derived sub-systems intelligible to all, any defense planning system requires a definition of "organic leadership".

Organic leadership is to be understood as a systemic network, consisting of a strategic goal-setting, operational planning, tactical implementation (including C4I), leadership in a narrower sense, management/administration and process-control within the armed forces. Each individual component includes (a) a combination of leadership qualities for goal-setting, planning/evaluation, command and control, human-

oriented leadership, target/actual comparison, and (b) specifications for the different echelons respectively process steps of the defense planning system in a broader sense (military-strategic, operational and tactical level).

Leadership qualities are processed in interactive cycles within each echelon and, at the same time, is the sequence of evaluation steps that are assigned to the different hierarchy levels, executed in a separate, overarching iterative cycle. By this simultaneity of different autonomous process-cycles, can one of the core principles of the generating defense planning system be defined and explained.

Pursuing the hypothesis that a generating system determines the derived sub-systems of defense planning, the constitutive principles of the parent-system and their functions can be identified. Subsequently, it will have to be analyzed in which way any of those principles might influence the development of sub-systems. Finally, it must be answered how all these elements can be aligned and controlled in order to achieve planning/C2- superiority by means of implementing them at all levels of the sub-systems.

Fig. 2. Interrelation "Parent-Defense Planning System" – Derived Sub-Systems

4 Analysis of the core functional principles of a parent-defense planning system[6]

4.1 Functional principles based on history of thought, military leadership-philosophy or other military/leadership-scientific foundations

The core of fundamental functional principles is mainly formed by those who focus on the creation of the generating parent-system of defense planning out of history of

thought-related, leadership-philosophical or military/leadership-scientific findings. As they are mostly abstract and quite generalized, they would apply to almost all derived system processes without much adaptation. Additionally, they represent basic principles that were barely subject to ongoing historical changes.

The separation of the strategic and operational level: This principle represents a bundle of basics of strategic leadership which jointly serve the overarching goal of relieving the strategic level from too concrete short and mid-term planning and further execution-tasks; in turn, will this separation principle allow focusing on the creative quality of the strategic objectives. Regardless of whether it is about the leadership-theoretical dualism of philosophical deduction and pragmatic operating instruction, or about the differentiation between creative phantasy and critical-analytic thinking within the circular strategic planning process - all these functions aim at "liberating" the strategic level from the constraints of technicalities. Checking identified options for feasibility already in the phase of strategic goal-finding, must lead to a loss of creative quality.

Of course, the feasibility has to be checked, whether by *one and the same staff element* in a subsequent separated phase or by a *different staff element* which is responsible for operational planning and the implementation of strategic goals. The latter variant which uses different staff elements requires a system of simultaneously conducted, autonomous planning-processes on all echelons of military leadership, as explained above.

General military leadership doctrine provides an approach to cope with the demands arising from the dialectic relation between goal-setting and feasibility-check within the strategic planning cycle. It suggests an iterative change of planning-directions from "visionary to pragmatic" to "pragmatic to visionary", based on the hermeneutic-circle principle which was developed by Clausewitz. As can be shown, the visionary power of strategic goals will be strengthened by separating goal-setting from feasibility-evaluation. Military leadership doctrine describes the dialectic relation of the latter in form of a metaphor: Soldiers of a marching unit follow the polar star, which represents their need of the farthest possible orientation-point to guide them - although no marching soldier ever really wants to reach it. [7] (cf. De Spiegeleire, Trend 6. Breaking the Dictatorship of the Present). Because far-sighted strategic goal-setting, in most cases, requires restrictions (in the present), the overall logic of which is seldom comprehensible to the subordinate levels, human-oriented leadership is a bridge between strategic goal-setting and operational planning, and, by the same token, an effective instrument to gain the trust of the subordinates for the strategic goals.

This model of separation also fits private businesses, as they also contain strategic and operational levels – both providing the strategic management with the option to pursue intended long-term goals without being locked in immediate, creativity-hampering constraints of short-term (economic) success. What military science certainly will expect from the civilian research branches (besides technological early warning), are ideas for creativity training, preferably under stress conditions, and testing methods for finding or developing creative talent.

Dialectic interrelation between theory and empiricism: The Prussian military theorist Carl von Clausewitz developed a scientific method around a synthesis of Kantian rational a-priori-thinking, and the empiric critical-analytical research of military-historic experiences. This principle will be used as one of the cornerstones of the parent-system model: Its correct understanding would prevent planners from changing structures, due to (wrong) "Lessons Learned" that are not checked adequately for a systemic rationale of malfunctions. The battle of Cannae e.g. would then not be praised for its tactical brilliance anymore – it would much rather have to be taken as an example for choosing wrong strategic goals.

Application of operating instruction versus self-dependent evaluation: Given the required humanistic and military scientific education of military leaders, the qualification for any self-dependent evaluation - in combination with "mission-oriented leadership" - significantly raises the quality, effectiveness, and self-regeneration capability of any parent-system. Already Martin van Creveld found that German staffs in World War II could be kept much smaller in numbers than American ones, because they could build upon a professional independent-evaluation capability and were therefore able to do without a large number of pre-developed contingency and other more detailed plans.[8]

"Divinatory Component": Dealing with the qualification of military leaders for self-dependent evaluation, one will certainly have to tackle what Clausewitz called "Takt des Urtheils", "Coup d´oeil" or the "Divinatory Component". However, could the analysis of these principles easily lead to the fallacy that such qualities were based on God-given, magic, and almost innate ability. By contrast, effective military system planning has to start with the understanding that the above evaluation capability derives from rationally developed conceptual doctrines. These are taught, learned and internalized by frequent practice to make the leader think he acts out of intuition, although in fact, there is always a rationally developed evaluation logic in the background. Indeed, such evaluations, in the eyes of outside observers, may lead to the impression that decisions were taken intuitively.[9]

Operational creativity versus critical analytic assessment: As the analysis of military operations and experience from operations and tactics proves, failure and defeat is mostly due to the lacking ability to recognize other (hitherto unnoticed) options. Military science has therefore started to develop the doctrine of system-generation, based upon the ability for operational creativity.

In this context, Professor Bernd Rohrbach (lecture at the 14th General Staff Course/NDA Vienna) assigned leadership capabilities to the "operating modes" of the human brain. Typically, each of these qualities hinders the other, when called up simultaneously. According to the different individual capacity of staff members with regard to operational creativity and to critical analytical thinking, Rohrbach identified a specific talent for "inventions" or for "management tasks". Logically, a well balanced ratio between both qualities will predestine a person as a possible superior leader. In case decision-making processes have to be carried out by one person alone, Rohrbach ("Deferred Judgment") suggests to blank out at first the rational mode (in order to have all brain-capacities available for finding options) and, in a next step, to suppress the creative mode (in order to raise the critical-analytic assessment quality).

Mission-Oriented Leadership: This principle provides the basis for extending the overall creative system-performance through decentralization of execution. It allows for using the mental capacity of all subordinate levels to achieve the overall goal, but in contrast to the principle of all "Powers to the Edge", by strict target-orientation in both directions, i.e. by methodically specifying “room for maneuver” (top-down), and by supporting the self-harmonization of subordinate levels, while at the same time encouraging and permitting creative inputs going from the bottom to the top (“bottom-up” process).

Military Scientific Basis: Military science will have to provide the required scientific quality in the core-subjects of the military leadership doctrine. Only this, not yet sufficiently completed scientific expertise in the *core-subjects* will allow for a harmonized synergetic input from the side of military scientific *accessory subjects* in the fields of human and social science (in particular military philosophy), military history and natural sciences.. Defense universities, in close cooperation with military and civilian research and development elements, will have to act as backbone for broadening and deepening the military scientific basis.

Especially in a period of declining defense budgets, it has become increasingly necessary to preserve and develop collective system-knowledge (“soft skills”). That might be comparable to a tree that throws off leaves before winter and collects its lifeblood in the trunk, in order to be able to sprout again next spring.

Code of Values: Systemic Defense Planning will also have to take into account aspects of values, because the overarching goals like “primacy of politics”, “campaign and mission-effectiveness” will require a code of values which necessarily must differ from those of civil society. A deepened understanding of values is indispensable for the trust of the subordinates in strategic goals and thus, in the benefit of separation between strategic and operational defense planning level. A reliable code of values is also inalienable in order to immunize all echelons against unlawful orders and thus, to ensure the primacy of politics.

4.2 Providing the basis for the structural/procedural alignment of derived sub-systems

The functional principles tackled in this paragraph must be seen as an interface between the core of the parent-defense planning system and the derived sub-systems. As such sub-systems are less constitutive for the functional logic of the parent-defense planning system, they are, for reasons of limited space, only mentioned here but not analyzed in depth.

Process-orientation instead of divisional-organized structures: Process orientation favors the holistic development of skills under horizontal cross-viewing of the different services, branches of services, personnel/materiel/infrastructural/educational aspects of any addressed capability development (EDA provides the role-model for such a structure).

Generalist-oriented role profile of military leaders: The general staff role profile for military leaders will gain in importance for creating and controlling process-oriented planning systems.

General standards for derived system-planning processes: These standards comprise specifications regarding the iterative character of planning processes, their periodicity/pulse-frequency, and the rhythm of change between their visionary and pragmatic planning phases. The above standards represent the essential control instruments for the definition of the sequence of goal-setting and feasibility assessment steps and for the ratio between planning range and rotation frequency (of planning loops). They are the basis of all subordinated processes.

5 The final result: "A suggested model of a parent- defense planning system"

The product of the proposed research is a proposal for an (abstract) next-step model of a parent-defense planning system. (See figure 3).
This model is based on functional principles, derived from history of thought-related leadership philosophy or acknowledged military/leadership-scientific findings. It is designed (a) for the regeneration of its own functional logic and (b) for the generation of the sub-systems' strategic planning-cycle (e.g. threat assessment cycle) capability planning cycle, procurement cycle, operational planning cycle and the tactical decision making processes.

Fig. 3. Model of a "Parent-Defense Planning System" to be proposed

Due to its level of abstraction, it can also be used for strategic and operational planning processes in the private sector or in the (security-) political decision making. The model is constructed as a dialectic synthesis of the antagonistic pairs of factors as analyzed above.

System decisions (regarding e.g. a transformation of systems, the creation of new structures or processes) as well as planning/C2 decisions are to be made in an environment that comprises two different intellectual domains: One of these domains is characterized by theory-heaviness (caused by philosophical-rational thinking) and operational creativity, while the other domain is based on the antagonistic pillars empiricism (systematic empirical research) and critical-analytic evaluation.

The evaluation process usually starts with the finding of usable principle options. Such is normally done by a creative staff cell, consisting of e.g. J2, J3/Plans and J5 (NATO-nomenclature) within the theoretical/creative domain. The Chief of Staff, whose responsibility it is to guarantee continued coordination and harmonization of creative and analytic staff cells, assists the staff with short-listing those options that are eligible for a working hypothesis.

The commanding officer (CO) will then decide which (first) working hypothesis is dealt with in the first loop. As explained above, he bases his decision on an experience-based, but at the same time, a priori rational evaluation ("Divinatory Component").

The choice of the working hypothesis is by no means to be understood as an anticipated decision that only has to be "legitimized" by the "analytical" staff cell. By contrast, it is the task of this analytical staff elements to try as hard as possible to falsify the hypothesis (see Karl Popper`s "falsification" theorem and "induction-problem") – as any failed attempt to push the hypothesis "out of the saddle" will increase its validity.

The analytical staff cell takes on the role of the enemy/competitor/alternative option, and tries to falsify the working hypothesis by critical/empirical methods. In case there is a definite reason for doing so, the analytical staff cells might discard a chosen hypothesis; otherwise the latter crosses the boundary to the creative rational domain. There the "creative" staff cell decides if the checked hypothesis will be stored in the "not possible to falsify-basket" (and will have to wait for the final decision of the CO) or whether it has to be modified and sent into the next loop again.

Finally, the CO, who is solely responsible for both intellectual domains, takes the decision by choosing one of the options that have already passed the evaluation cycle successfully. This assessment is based on his experience and at the same time on rational thinking.

Thus, this approach combines the advantages of the empirical and the theoretical science-paradigm and also provides creative, far-sighted goal-setting capability, underpinned by an independent, unerring critical feasibility analysis. Such process is finalized by an "acid test" under conditions of practical implementation.

Notes

1 Defense planning is understood here „in a broader sense“, i.e. as the overarching planning system, including defense planning in the narrower sense, defense procurement and operations, whereas defense planning in the narrower sense only would comprise fundamental planning, research and development and capabilities-/force planning.

2 Cf. Paul J. Di Maggio and Walter W. Powell: The Iron Cage Revisited - Institutional Isomorphism and Collective Rationality in Organizational Fields Author(s) in American Sociological Review, Vol. 48, No. 2 (Apr., 1983), pp. 147-160, Published by American Sociological Association Stable.

3 Cf. Stephan De Spiegeleire (2011): Ten Trends in Capability Planning for Defence and Security, The RUSI Journal, 156:5, pp. 20-28.

4 Any deeper analysis that aims at the creation of a complex meta-planning-system will have to include the findings of Kent Palmer, who builds upon Heidegger and Merleau-Ponty (Cf. Palmer, Kent (2000): Reflexive Autoieptic Systems Theory http://works.bepress.com/kent_palmer/). However, had the philosophical basis of this research - due to the required brevity – to be limited to Clausewitz and in particular to his dialectic antagonism between theory and practice..

5 Cf. Stephan De Spiegeleire (2011): Ten Trends in Capability Planning for Defence and Security, The RUSI Journal, 156:5, 20-28, trend 3 “Towards Capability Portfolios Based on New Partition Schemes”; the ten trends analyzed in this article are suitable as checklist for the overall effectiveness of planning systems - this is why the suggested ideal-typical model was to be evaluated against the backdrop of these ten trends – following quoted as „De Spiegeleire, Trend 1-10“.

6 The assessment in this section follows - with respect to specific items - to some extent and in a figurative sense the analysis of military functional principles in the draft-thesis, which is submitted by the author, to the National University of Public Service/Faculty of Military Science and Officer Training, Budapest (title: Relations between Functional Principles of Democracies and their Armed Forces, 2013).

7 Cf. Hans H. Hinterhuber: Die 5 Gebote für exzellente Führung, Wie Ihr Unternehmen in guten und in schlechten Zeiten zu den Gewinnern zählt, F.A.Z.-Institut für Management-, Markt und Medieninformationen GmbH, Frankfurt am Main 2010, ISBN 978-3-89981-228-2, p. 63.

8 Cf. Martin van Creveld: Kampfkraft, Militärische Organisation und Leistung der deutschen und amerikanischen Armee 1939-1945, ARES-Verlag, pp. 65-71.

9 See e.g. the intuitive and offensive qualities of Field Marshal Rommel. Since he never attended a general staff training course, he lacked a number of essential operational skills, like adequate defense tactics and the importance of logistics.

Turning a Suite of Modeling and Processing Tools Into a Production Grade System

Pascal Rivière[1], Olivier Rosec[1]

1 Caisse nationale d'assurance vieillesse (Cnav)
110 – 112 avenue de Flandre, F-75019 Paris
pascal.riviere@cnav.fr, olivier.rosec@cnav.fr

Abstract. The French national agency for old age pensions has evolved, along a systemic vision of all the issues to be covered, a generic set of tools to design and process structured file formats. This set of tools has turned into a production grade modeling and validating suite which relies for its distribution on the continuous integration of its various components. As such, continuous integration has here turned into a system which industrializes the production of systems of tools, each characterized by the file format it implements and the version under which it is released.

1 Introduction

Social security agencies have to rely on a massive influx of data to assess benefits and collect contributions. The data comes in thanks to structured file formats. Processing and validating these files is not easy, because of the many changes in their specification each year.

This is why the national agency for old age pensions (Cnav) wanted to avoid rewriting endlessly its file processing applications, and wanted to maintain a constant level of processing quality. A general approach was necessary.

To achieve this, the national agency has elaborated a generic set of tools [Rivière, Rosec 2013]. It was initially used on the N4DS standard, and then on the new DSN standard (N4DS = Norme de Déclarations Dématérialisées De Données Sociales, DSN = Déclaration Sociale Nominative).

This approach can be applied well beyond the initial problem domain: collecting data from payroll systems.

The suite of tools has been improved and extended along a systemic view of all issues to be covered when dealing with structured file formats: design, validation logic, testing logic, documentation both of the file formats and the semantics of the business data they carry.

2 What is an interchange standard?

Elaborating a generic set of tools for modeling interchange formats and for generating the corresponding validating components brings back to the original issue: defining a metamodel for an interchange standard. We define it by the following characteristics:

- It should implement a conceptual model of the business domain;
- It can be derived into a series of message models carrying the business data payload
- Each message is a hierarchy of data blocks which respect a certain structure and multiplicity (0,*)
- The structure of a block is defined as an ordered series of data elements which respect a certain syntax and multiplicity (0,1)
- The value and syntax of a data element is defined by a business datatype restricting one of four technical datatypes: string, decimal, enumeration, external referential
- Each technical datatype has facets which help define business datatypes: minimum and maximum length, a regular expression for string and decimal etc.
- Semantic consistency between data elements is enforced by rules.
- Because the interchange standard has to support the interchange scenarios defined by its user community, message models can themselves be inserted between a header and a footer composed of blocks linked to the applicative logic of the actual exchange system.

On top of this metamodel one can model many interchange formats. Here are the metrics for a few of the interchange formats modeled with the suite of tools.

Number of	*N4DS v01x08*	*DSN phase 1*	*DSN phase 2*
Data blocks	*124*	*29*	*33*
Data elements	*687*	*202*	*258*
Validating rules	*995*	*198*	*210*

The DSN message is the first for which a conceptual model was elaborated before working on the corresponding file formats. The idea is to derive the message model from the class diagram (below). It should be noted that such a derivation cannot be 100% deterministic and requires some choices [Elmasri, Navathe 2011].

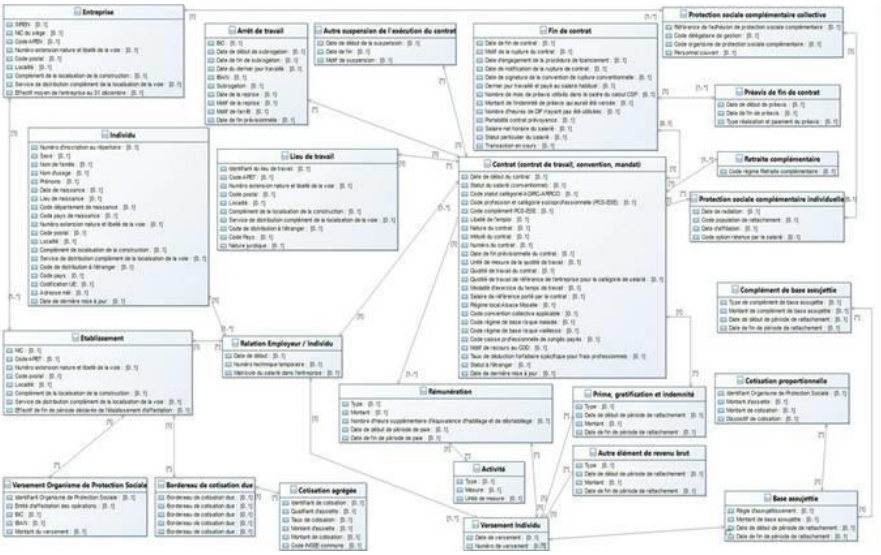


Fig. 1. DSN phase 2 class diagram

The figure below shows how various message models can be inserted between header and footer blocks.

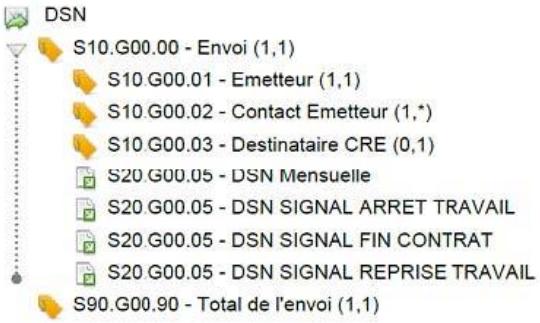


Fig. 2. DSN phase 2 Message Hierarchy

The figure below shows how the biggest DSN message model, the monthly DSN, is represented as a hierarchy of data blocks in the Designer.

Fig. 3. DSN phase 2 Monthly Declaration Hierarchy

Each data element is described in detail in the .pdf export of the specification.

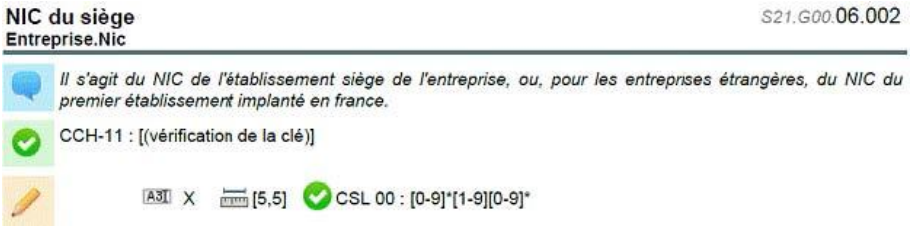


Fig. 4. PDF export for a data element

3 Tools, what for?

The generic set of tools covers the following functionalities (corresponding tool between brackets):

- Modeling an interchange format and defining its validating rules (Designer)
- Documenting the data carried by the interchange format (Semantic Repository)
- Generating the specification of the interchange format as a .pdf + various .csv exports (Designer)
- Generating validating components which can be integrated in a process (Parser within the Designer, Validating building block composed of Knowledge Base cum Validator)

- Generating a validating tool with a graphical user interface for a local compliance check before sending files to the remote processing platform (Autovalidator)
- Testing the generated components (Tester)

This series of tools does not owe its birth to a careful plan. One should rather think of it as an emergence. It is through the efforts of the national agency to organize the original system of goals and means and the various artefacts supporting them into a coherent whole that new tools kept arising until the present list was obtained.

How do these tools operate? This is what this part of the paper is about.

3.1 The Designer

This tool is at the heart of the suite. As previously noted, the Designer generates:

- The specification of the interchange format which will be disseminated throughout the user community for implementation (.pdf, .csv export, XML schemas)
- But also the knowledge base which will be read by the Validator to check whether the actual files sent by the users comply with the specification

As a modeler of data structures, datatypes and messages, the Designer is fairly classical. But as a repository of validating rules relying on first-order predicate logic, it is not.

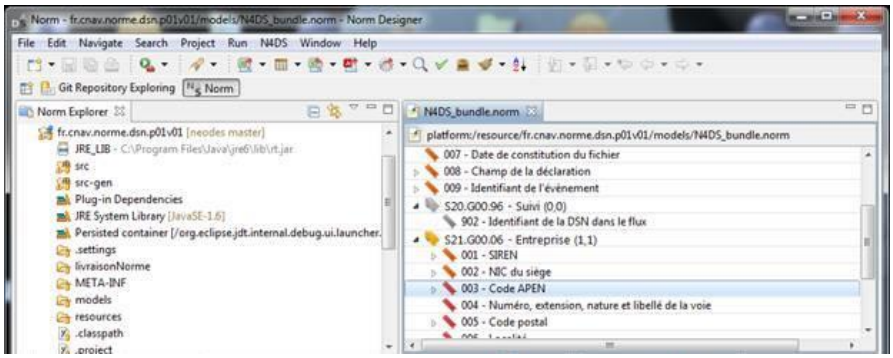


Fig. 5. Top part of the Designer UI

In the modeling perspective, the screen is split up between the project view (top left), the outline view of the current model (bottom left), the view displaying the current model element (top right) and the view showing the properties of the part of the model currently in focus (bottom right).

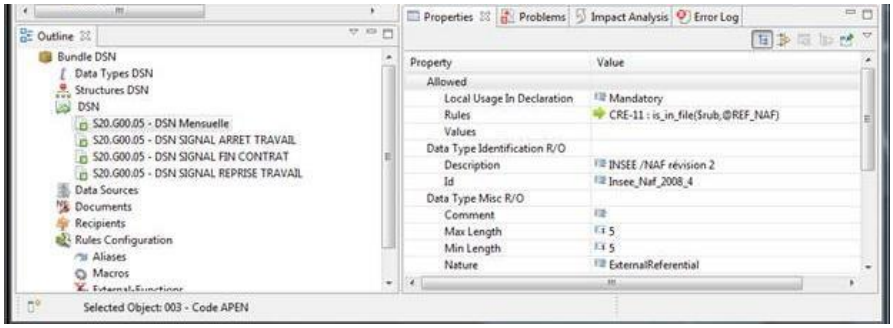


Fig. 5. Bottom part of the Designer UI

Rules are modeled under the data element to which they are attached in the functional specification for the file format.

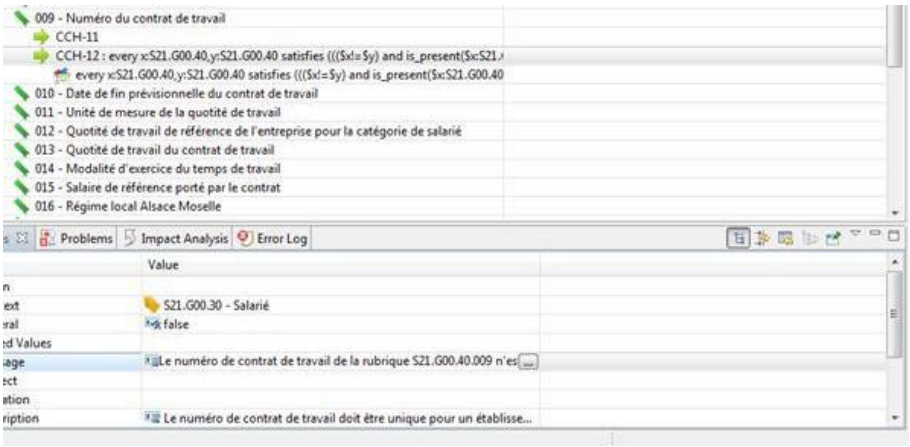


Fig. 5. Rule as displayed in the Structures and Properties views

A rule has a subject and a context, that is, a position within the message tree from which all paths to the data elements called by the rule will be calculated. The textual DSL corresponding to the functional rule is entered in the model through a widget.

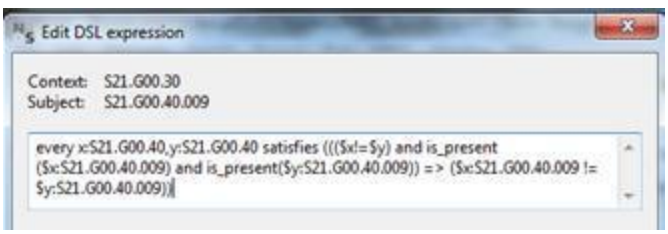


Fig. 5. Rule as displayed in the DSL widget

A message will be returned to the user in the report stream if the processed file does not satisfy the rule.

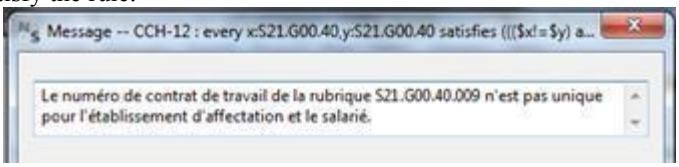


Fig. 5. Rule user message

When the knowledge base is compiled, the rule will be implemented as a Java class along with the set of utility classes loading and unloading at runtime the data elements called by the rule.

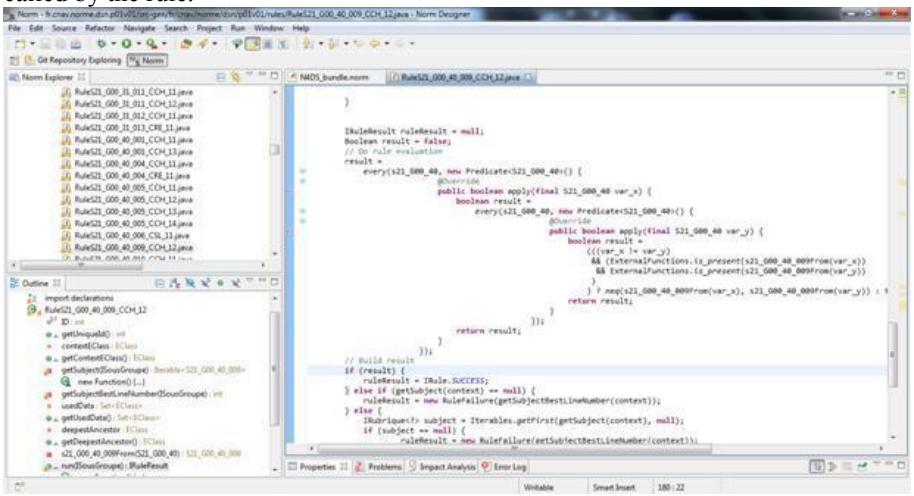


Fig. 5. Java implementation of a rule

3.2 The Validator

When a file is processed by the Validator, it goes through several stages:

- Conversion into XML
- Syntactical validation
- Semantic validation

As the file is processed, the Validator emits a report stream. The screenshot below shows the console and part of the report as transformed from XML to HTML by a local script.

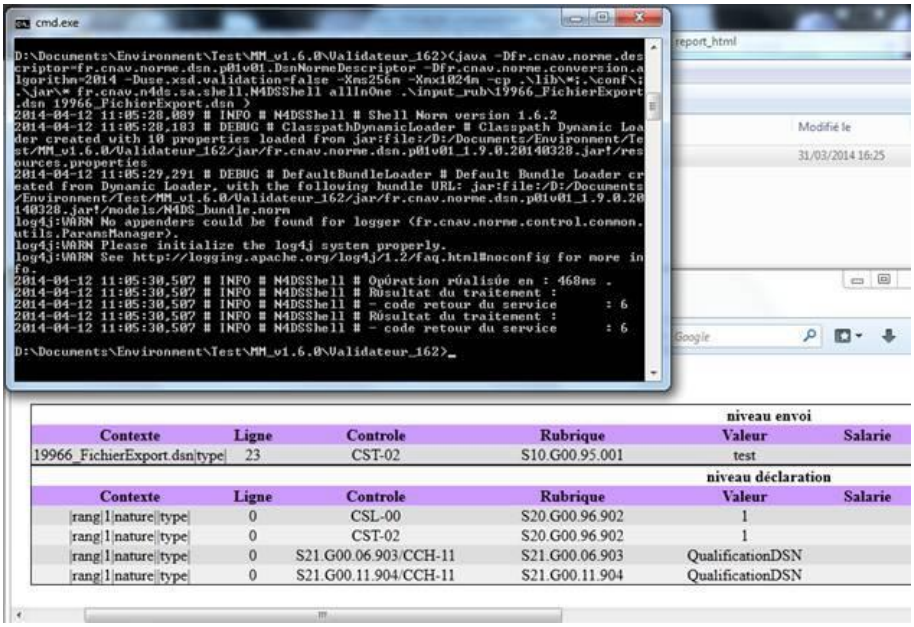


Fig. 5. Command line view of the Validator against backdrop of HTMLized report

The Validator has been improved: the report stream can be customized to the needs of any user community, thanks to an architecture which now separates events logging and report serialization

The Validator has been rolled out in production for the DSN.

A POC has been made to validate file formats circulating between old age pensions agencies to consolidate data about pensioners claiming the minimum entitlement.

3.3 The Tester

Work is under way on a testing tool which should:

- Validate the initial knowledge base against the specification it implements
- Ensure non-regression on validating components before roll-out
- Generate semi-automatically test suites.

The Tester is an RCP tool which loads a test suite file into a series of test cases which are then automatically executed and parsed to compare the obtained result with the expected result.

3.4 The Auto Validator

Users can now check whether their files comply with the specification before sending them to the processing platform.

The Autovalidator tool reads the same knowledge base and executes the same validating logic as the Validator in production.

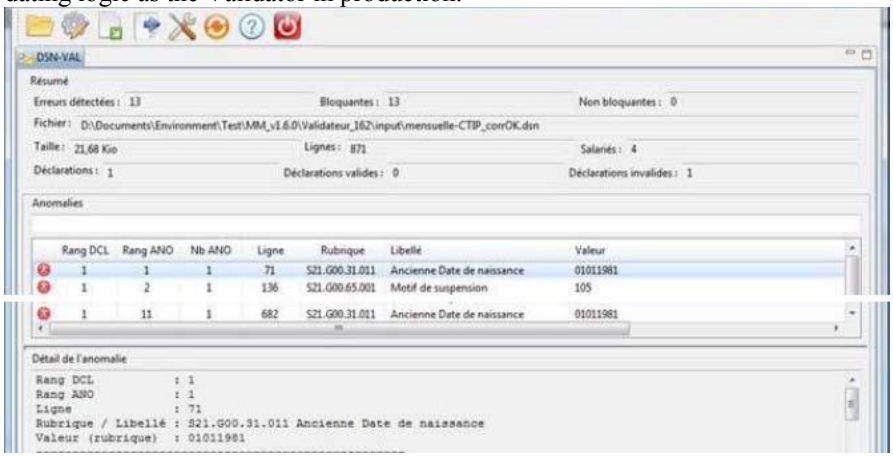


Fig. 5. Screenshot showing local validation with DSNVAL tool

Anyone can download the Autovalidator tool for the DSN, DSNVAL, on the Internet: http://www.dsn-info.fr/precontrole-dsn-val.htm

3.3 The Semantic Repository

A Semantic Repository now links data definitions coming from the conceptual data model of the business domain to their implementation in the file format specification. A query tool helps navigate through the Repository, using various approaches: selection criteria, full text research, keywords...

Fig. 5. Welcome page of DSN Semantic Repository

4 Implementation.

All the tools call on the facilities offered by the Eclipse Modeling Framework.

The Designer is built on top of the Eclipse platform. An effort has been made to organize its plug-ins into coherent features. It has moved from a mere graphical modeler to a developing tool which federates into a specialized plug-in all the projects an interchange format relies on for its successful implementation: modeling projects, and Java projects too in the guise of the functions extending the textual DSL, the few remaining rules written directly in Java, the Java serialization for the report stream.

A class known as a Norm Descriptor calls the validating logic persisted in the knowledge base. The knowledge base itself is a Java archive organized in layers according to the processing stages of the Validator.

In the models layer, one finds the files which describe the syntax of the model the actual files have to comply with, in terms of data blocks, elements and types.

A Java project carries the logic for the semantic validating rules and the report serialization.

Report logic relies on a model which is shared by the Designer and Validator.

The conversion algorithm which transforms the input files into XML has evolved at the beginning of 2014 to throw an error when a data block or element is out of sequence with regard to the “covering message” (the superset of all message models for a file standard). It thus contributes to the validating logic. Previously it generated extra

data blocks elements to generate a sequence of XML elements which would always be valid, but this strategy led to misleading error reports at the syntactical stage.

This is the most recent change brought about by the need to bring the Validator to production grade level, after the replacement of the initial off-the-shelf engines. The semantic validation was ported from Saxon and Schematron to a Java API in 2012 for performance, and the syntactical validation was ported from Xerces to the same Java API because users did not understand report messages phrased in XML constructs.

5 Continuous integration.

Originally, continuous integration covered only the suite of tools which were then more or less used on a standalone basis by the unit which specializes in the generation of the format models and validating components within the national agency.

But as the Validator was being prepared for production, it was considered safer to include in the testing strategy the knowledge base itself.

Now continuous integration encompasses all source control repositories, whether for tools or file formats, and builds to roll out fully tested components can be specialized according to the phase and version of the file format for which deliverables must be shipped.

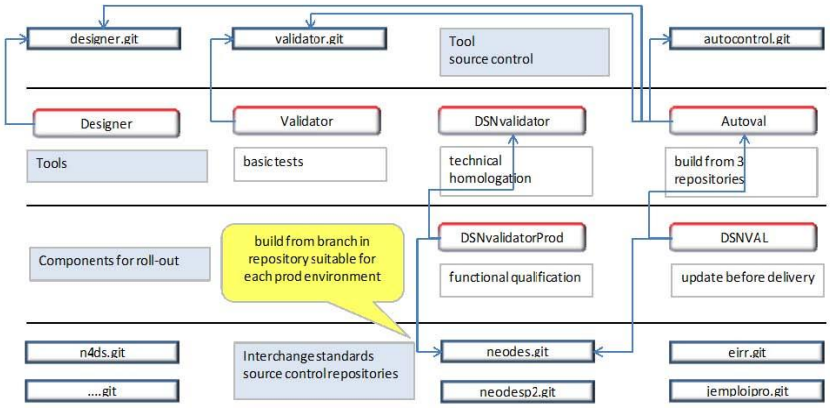


Fig. 5. Code repositories called by builds according to tools and deliverables

Continuous integration has become a system which helps produce systems of tools for structured file formats, each system of tools and deliverables being specialized into a given file format and released under a certain version.

Genericity still prevails thanks to the separation between models representing business rules and transformations implementing the processing logic.

One could almost describe the system of code repositories and build scripts as an autopoietic machine perpetually regenerating tools and deliverables.

6 Conclusion

The suite of tools is used intensively and successfully for the DSN project.

The national agency wants to make it yet again more generic in order to generalize its use for the validation of most or all structured data interchanges between other institutions and itself.

It is also investigating the possibility of turning the suite of tools open source in the hope it nurtures a new way of thinking about and working with structured file formats.

References

1. Pascal Rivière, Olivier Rosec, “Model-Based Interchange Formats: a Generic set of tools for Validating Structured Data against a Knowledge Base”, CSDM 2013
2. R. Elmasri., S.B. Navathe, “Fundamentals of database systems”, Addison-Wesley 2011

If We Engineered Systems Like We Produce Movies

Dominique Luzeaux[1], Thierry Morlaye[2], Jean-Luc Wippler[3]

[1]Deputy Director of DIRISI, Ministry of Defense, France
dominique.luzeaux@polytechnique.org

[2]President of COERENSIS, Paris, France
thierry.morlaye@coerensis.com

[3]Owner of LUCA Ingénierie, Toulouse, France
jlwippler@gmail.com

Abstract If we engineered systems like we produce movies, would our enterprise and endeavors prove leaner and more agile? We daydream on the greener pastures of movie making, an industry whose signature practices were born out of necessity, to adapt to major regulatory upheavals in the 30's Hollywood, and have since proved remarkably effective & resilient. We journey through odd ways of tackling strangely familiar problems, subtly different organizational patterns, and seemingly unreasonable, yet strikingly efficient practices. Could we gain some insights or fresh ideas, to renew and better up our own 'complex system engineering' practices?

Keywords Processes, People, Lean, Agile, Movie Making, System Engineering

1 Introduction

Is a movie a system? Of the kind we routinely design & develop in our 'complex systems' industry? Such a discussion would probably prove entertaining, and possibly shed some semantic light on our daily struggle with system modeling. Yet, we choose to focus our own investigation on a different point of view, of a teleological rather than ontological nature.

There are at least two goals that are worth considering, and could be deemed as 'equivalent' in both worlds:

- Design and sell (or provide) a profitable product (or service). Obviously, movie producers invest money so as to gain strong return on investment in the process. With direct control of their stakes, they have strong incentive to make things right, and they generally have succeeded, with superior return and gains, despite some odd, spectacular failures. While profitability is certainly a goal we share in 'our

part of the world', we often fall short, as so clearly demonstrated e.g. in the software industry [2].
- The success of the 'End Product' lies in some 'good properties' that reflect a sound architecture. It would be a bit conceited to actually synthesize what makes a good movie in the same way as Vitruvius did a long time ago for Architecture[1] [3]. Nonetheless, 'good' movies are actually architecturally conceived, and we should look into this process.

As a preliminary, let us remark though that many modern movies or TV series are endeavors on a scale rivaling many traditional 'complex systems', involving millions of dollars spent each week, with tight, imperative schedules (24 episodes each year for a TV series), hundreds of contributing people, in distributed teams located in many countries & locations.

'How movies are made & produced' may appear, at first glance, quite alien & estranged from the way we classically engineer systems. After all, in his thought-provoking book [12], Goranson traces the key inspiration of many of these practices to the experience of the… Nantucket Whalers! On a closer look, though, we pick hints & ideas that resonate with our own daily issues in system engineering:

- Modern movies often have a complex lifecycle, as they are distributed through a growing variety of channels, from theaters to TV broadcast, DVD and video tapes, on-demand pay-per-view, in various markets, timeframe, formats, countries and languages.
- Shifting webs of relationships link stakeholders commanding these various channels, with opportunities (or regulatory requirements, e.g. in France) of complex risk & profit sharing schemes, cobranding and joint promotional efforts, including toys, books, music, sequels, prequels, derived comics, and all the paraphernalia of modern marketing.
- These result in significant up-front investments, either in concept definition, actual shooting, post-production, marketing and distribution. To perform these tasks, the industry enlists a bewildering variety of highly specialized companies and individuals, from unique special effects (FX) teams to dedicated insurers, or apparently mundane but key, specialized carpenters[2] or dedicated catering.

[1] According to Vitruvius, a 'good' architecture shall meet the following *Vitruvian set of three*: ***firmitas*** (solidity or robustness), ***utilitas*** (convenience or usefulness), ***venustas*** (beauty or sensual delight).

[2] Goranson [12] gives an enlightening example with the shooting of the movie *Waterworld*, which required building up floating sets. Instead of trusting highly-paid & specialized carpenters & movie professionals with developing these sets (an expensive but sound, $3M solution), the studio decided, inspired by its new, novice Japanese owners, to pick up subcontractors specialized in sea-faring buildings. In so doing, they failed to capture the innumerous, implicit knowledge & language of movie-making professionals that, in so many crucial details, is critical to delivering movies on time, and in having all specialized teams dance in tune. The implied delays ended up costing about $80M…

- Already in the 30's in the United States, the movie industry was highly concentrated and competitive, leading them to adopt such "modern Lean" practices as flat organizations, just-in time or prequalified suppliers [12]
- Today, movie production is still a living example of "Lean" principles in action. Indeed, an obvious keystone of movie production is that customers define value: every movie aiming for profit focuses on a specific audience, and tackle to its expectations. Producers devote extensive care and critical attention to the project early on, studying it at length, getting feedback from numerous and various contributors, formulating 'the right project' before committing shooting resources, looking for an overarching and generative 'High Concept' [14], a fitting cast and crew, committed financers and distribution partners, and extensive research on potential risks, Intellectual Property and regulatory issues.
- Actual shooting will only proceed once pre-production has achieved a clear, shared vision of the project, targeted audience & success factors, and once potential risks have been properly identified, accounted for and mitigated. Shooting will then proceed at an amazingly fast pace, with the innumerous specialized contributors 'magically falling into place', seamlessly cooperating to shave out unnecessary delays or unplanned rework out of the value stream. To this end, scenes are shot out of order, some stars and actors will only appear on stage for a couple of days, sets will be prepared in parallel, and so on. Clearly, time is of the essence, the main cost driver at this stage, and the whole process is ruthlessly optimized accordingly.
- Actors on a stage may spend most of their days idly waiting, with a few seconds of shooting randomly interspersed, a seemingly shocking lack of productivity for high paid, otherwise demanding stars. For the clear focus of the whole production is not on the specific productivity of individual assets, but on ability of the whole production to deliver quality on time.
- Another feature of movie production is agility, in the sense of agile engineering. Although there are processes for making movies, directors will more likely interact directly with actors. They will also react in face of change (e.g. unexpected weather changes, star actor with food intoxication, or grossly out of shape, as Marlon Bando joining the shooting of *Apocalypse Now*), adapting on the spot, turning planning ordeals into artistic opportunity. And obviously the emphasis is on making a fitting movie, i.e. a valuable end product, rather than on generating plethoric documentation[3]. As a matter of fact these are the fundamental values of lean, agile engineering.
- Actually, documentation and support processes are not absent from movie making, and play a key role, e.g. when a script girl expediently notes, with amazing accuracy, all details of a scene. But these roles only exist insofar as they are critical to the quality and performance of the whole process – and, as such, are granted as much care and respect as other roles & tasks.
- In addition, many producers and directors will actively engage with the targeted audience throughout post-production, testing the movie on smaller focus groups.

[3] Although the paper edition of the scenario is part of a movie's collectibles, it is obviously not the purpose of the movie-making process!

This guides editing and cut, a critical step to tune the pace, tone, sense and impact of the movie, and to fit the music score, FX, rushes, and numerous other parts into a cohesive and expressive whole. Actually, in a 'design for flexibility' kind of way, the director will often take a few extra shots of a scene to hedge risks, explore alternatives, and gather supplementary material (in a cost efficient way), so as to enable 'real options' later on, at cutting stage.

- Obsession for 'the right product' thus percolates throughout the whole project, from the pervasive front-end exploration to the post-production and distribution stage, combining a process-oriented mindset (shooting) with serendipity and attention to key 'emergent properties' afforded by the consistent efforts of many, varied, specialized, cooperating talents.

2 New roles and organization for the system project team

2.1 From duo to triumvirate

Complex system projects are often lead by a duo: the project manager (aka director, or leader), and the system engineer (or system architect, or chief engineer, etc.), splitting *de facto* the project into two distinct, overlapping domains: engineering and management. There are numerous debates on who leads whom, who drives whom, who decides what, and so on. Actually, in our own, humble opinion, there is no convincing consensus yet as to the roles and the responsibilities of these two key players. Even the names of the roles are not so clear, as they keep changing through time & space.

For example, it is quite frequent in system engineering primers, to present the system engineer as a conductor of orchestra, a leader, enabling multiple disciplines to work together in a balanced and fruitful way. But a project leader claims also to be the leader… of the project (of course). And the importance of the composer, i.e. the one who thinks and designs the masterpiece to be then executed/realized, is often lost.

Let us imagine that we have both these distinct roles, as in music or in movies:

- The composer, or the writer, who we shall call the *System Scenarist*. He understands the complex problem to be solved, by taking into account the voices of the customers and the many potential stakeholders; he finds the solution, investigating and exploring numerous alternatives in the process; he generates, converges on and selects an efficient concept, capturing and defining the solution, based on his experience, the accumulated knowledge of its enterprise and, obviously, his own talent.
- The conductor, or the director, who we shall call the *System Director*. He leads the numerous people involved in the system development[4] process.

[4] We should take here an extended meaning for 'system development, not restricting it to the implementation process (as in ISO 15 288, or SE Handbook) but encompassing all the technical processes.

Note that both of them require a solid technical, scientific, mathematical (and so on) background. In other words, both are engineers, their differences lying rather in their behavioral skills and focus: one is leading, while the other is creating.

To 'achieve' successful movies, i.e. loved and cherished by the audience (the end customer, shall we say) and generate superior profits, we need, in fact, a third and critically important player: the *production*.

Quoting Film Training Manitoba [11], *"Producers are the driving force behind any project. They are essentially the 'managers' of a movie production. […] Many producers also have extensive experience in many aspects of filmmaking and have worked in many different positions on film or television projects. […] Producers secure investors and put together the financing for a production and they are ultimately responsible for all financial aspects of a film. They must have strong business management skills, […]."*

So let us add this key role to the two previous system roles:

- The *System Producer*, or perhaps shall we call him '*System lifecycle and profitability Manager'*. He will be in charge of the financial and contractual aspects of the system production all along the system lifecycle. As in movie production, he shall have a significant experience and a deep, practical knowledge on 'how to perform system engineering', being able to support the development team and give sound advice. In other words, not a 'yet another Excel sheet holder', but someone who is really, deeply involved and committed to the process.

2.2 Process is good, people are better

Even if the various roles of the triumvirate (*System Producer*, *System Scenarist* and *System Director*) are distinct and complementary, ensuring a 'per design' good balance in the leading team, much of the success lies in their actual interplay: how do they fit together, do they work on good terms, with mutual understanding and respect?

To these three core pillars of 'a movie-oriented system production', we should add, and not neglect, a complementary function that considerably fosters the success of a project: the *casting*.

Spotting the 'right' actors, and the right *mix* of actors, is a major and defining feature of movies success.

It is for instance striking to learn how the initial casting of 'LOST' had such a strong influence on the TV show (and its unquestionable success) [5, 6]. Some characters were added during the casting because it was recognized that the auditioned actor could bring significant value to the show, leading the writers to challenge themselves. Many young, talented women were auditioned for the role of Kate, but, when it was the turn of Evangeline Lilly (a totally unknown actress at the time), everything stopped. It *had to be* her, and no one else. Even the emotional tension that appears to certainly emerge between her and the character of Jack, played by Matthew Fox changed radically the initial script. Jack was actually supposed to die at the end of the

pilot! In a true agile form, everyone adapted to exploit this opportunity, unleashing new waves of creative thought & insights in the process.

Casting, or building up the right team is not restricted to 'recruiting' the good actors, but is extended to all supporting disciplines. A movie director will certainly strive to surround himself with his favorite cinematographer or his favorite artistic direction, according to the specifics of the project. These key players will come together around a specific project, around a common *purpose*. And, in a 'fail fast' kind of ways, early discussions between the key players may reveal irreconcilable takes on the project, leading some directors or star actors to quit the endeavor early, so as to open space for others that would better serve the project.

This leads us to question our usual way of assembling engineering teams. Too often we have seemingly blind faith in the process, as the one path to success and profitability. Since processes can be 'objectively' monitored through KPI (even the use of that acronym for key process indicators, so widely spread within the community, is almost a way of self-satisfactory reliance on ill-understood but smoothening habits!), they form a comfortable and lazy policy to rely on, and they lead to potentially *'Fail most successfully'* [13].

It is much more challenging to raise an actual policy of casting engineers, unleashing their combined talents to ensure success (as a pun to the reader, did you ever have a chance to do it in your practice?).

Quoting Ed Catmull, cofounder and president of Pixar: *"We believe the creative vision propelling each movie comes from one or two people and not from either corporate executives or a development department. Our philosophy is: You get great creative people, you bet big on them, you give them enormous leeway and support, and you provide them with an environment in which they can get honest feedback from everyone."*

Did we do otherwise when racing to the Moon?

2.3 Engineers do only engineering

Nowadays, we are often trivializing engineering trade. A spreadsheet, a slideshow presentation tool and an email client make up the basic, essential toolset for engineers, increasingly overwhelmed with dumb reporting, administrative & management tasks.

On the other hand, in movie production people seem to be more focused on their core job, relying on the support of specific assistants. E.g. the movie director has a first assistant to track daily progress, prepare and organize the schedule of the day, make sure that logistics follow up…

This may appear luxurious, but it ensures at its best that when a major decision is made, everyone is properly aligned, and that the decision is properly enforced. When Georges Lucas chooses among many raised propositions, for a character appearing only for a few seconds [8], then all follow suit. The management process behind that not only works perfectly, but it does not need him at all (by the way, e.g. configuration management in movie productions such as *Star Wars* is no small feats).

When turning on a profit is of such paramount importance to movie producers, their apparently paradoxical solution is to engage extra people, so as to free key play-

ers from administrative or management tasks. This is Lean in action: optimize globally, focus on the flow, ban rework whenever possible, continuously measure & improve…

All of this is in stark contrast to current common practices in our industries, where engineers have to perform all the management work, because of a patent lack of assistance, and, from time to time, quit their beloved reports to perform a precious few seconds of actual engineering!

Let us imagine how profitability could be increased if we re-focused engineers to do engineering and only engineering, supporting them with appropriate assistants to perform all the management and logistics tasks.

For example, a System Scenarist will mainly work on the creation process to 'discover' the optimal concept, exploring a broad playground, generating and assessing bunches of alternatives. For that, he will reason on high-level abstraction of the system under study, reflect on prior arts, failures and attempts, and provide probably more literary insights of the system to be realized. Then the precise iconic or graphical models needed for the system development (be it storyboarding or digital 3D mock-up in movie production, descriptive models like the various views of an architecture framework in system engineering) will be entrusted to some technical assistants, both human and digital.

3 New main lines for engineering a system

3.1 Life-cycle management and cost models

Movie making focuses on cost issues from the start on, and these concerns will infuse all processes of movie production. The movie industry actually developed eons ago what is nowadays called Activity-Based Costing accounting practices, long before the term was coined, or even the need widely acknowledged.

Indeed, before a movie project starts shooting, its funding must be guaranteed, with a clear business case, and investment commensurate to the expected revenues (i.e. coming from the theatre projections, the sales of DVDs and all sorts of collectibles related to the movie).

Translated into system engineering, this would imply that acquisition costs are determined relatively to utilization costs and revenues. Although this is the general rule for large-scale complex systems requiring public-private partnership investments (such as construction of new airports or major constructions such as tunnels – e.g. the Channel tunnel – or bridges – e.g. the Millau bridge –), it is clearly not yet an honored practice in many actual processes in the industry. Usually, only lip service is devoted to genuine global cost & value management. Indeed, although agreement processes (acquisition and supply processes) are explicitly mentioned in the 15288 System Life-cycle Process, their imbrications with the technical engineering processes is hardly described.

The way a movie production team develops *ab initio* the financial support of the movie could be also a source of inspiration for system engineering. This activity is clearly a key focus of the System Producer.

Lifecycle management for movies is a continuous process: teasers are progressively introduced on the commercial distribution market while the movie is being shot, carefully crafted stories about star actors are leaked, and once the movie is released, advertisements and commercial exploitation of the movie last for some time, being replaced by other kinds of commercial exploitation once the movie is retired from screens and begins its new life as a source of private projection revenues or diffusion on television channels.

Analog concepts for system engineering could be introduced within the stages preceding in-fielding, which would profit ultimately the utilization phase, where acceptance of a new product or service is not always easy, and is hardly anticipated by the acquisition teams.

Other issues deal with the end of the lifecycle: disposal of a movie is never actually the case, although some movies fall into oblivion, which could be seen as a state of definite retirement. However many movies become then 'classics' which is another way of recycling them.

Similar situations for systems engineering would be implementing sort of a circular economy process within the systems engineering process, which would be well adapted to the newest trends permeating our society in quest for sustainability. This implies building differently the basics of the current widely spread engineering way of life, which relies heavily on sequential activities and iterations of that sequential process, but cannot cope with circularity.

3.2 Project phases and progress oriented

Focused to the only objective of the movie opening and exploitation, movie making processes involve a set of widely acknowledged stages: development, pre-production, production, post-production and distribution/exploitation [9, 10]. Each of these stages has clear objectives, tasks to be performed, and 'definitions of done' as preconditions to next-stages transition. For example, the pre-production is all about preparing the shooting. It includes tasks like storyboarding, casting, selecting locations, designing and building sets, and so on.

Even if project phases, synchronized with system lifecycle stages are part of all system engineering primers & handbooks, most standards expand the central idea that performing a system engineering approach boils down to deploying a set of processes (25 according the INCOSE SE Handbook [4]). When engaging or training seasoned professionals, we routinely observe that precious little consideration is paid to how these processes have to be actually run and adapted, in tune to the project progress.

For instance, to mimic the movie production, there could be a 'phase 0'[5] while developing a system, when the concepts and the business case are jointly elaborated. This requires a strong involvement and commitment of the System Scenarist, the System Director and the System Producer. In other words, designing the architecture of the system & engineering costs are the two faces of the same coin, and imperatively

[5] Naming this phase more precisely would raise endless discussions at this stage…

have to be jointly developed by the core team, with a comprehensive view of the whole system lifecycle.

3.3 Everything starts with an idea.

Let us add a few useful aphorisms to the bunch of heuristics system designers should use:

- You should build your 'thing' around a central idea or concept. Use this idea or concept to build everything (it is generative).
- If you cannot explain your concept, your idea to your grand 'ma, forget it.
- Don't hesitate to throw an idea away, and start with a fresh one.

These heuristics originate from the architecture [15], and have consistently proved useful both in movie making and in system design.

In movie production, the pitch, even the famous 'high concept' [14], is the starting of everything, and at least the minimal, shared understanding that everyone (the cast and the film crew) have to 'get' so as to deliver consistent & convergent results.

For instance, the 'pitch', or the original concept for the sci-fi TV show '*Battlestar Galactica*' (BSG), was explicitly & emphatically not to produce "yet another space opera show with nice FXs", but instead to explore the concept of "putting the humanity faced to its own finiteness and its own creation", thus setting up a clear, open, dramatic and generative basis for narration, thus for 'designing the system'.

Actually, when BSG was under development, the production submitted to prospective lead actors a script of the pilot including, by mistake, the production manifesto explaining, in a couple of pages, these key design choices & concepts [7]. It completely changed the way the actors approached reading the script, capturing their imagination & will, generated creative energy, inspiration, and alignment, and drew to the show talented stars that would have declined it otherwise. Such is the e.g. case with Edward J. Olmos, of Ridley Scott's movie '*Blade Runner'* fame, who was clearly instrumental in the success of BSG development.

Nowadays framework in system engineering tends to 'explain' their system through a bewildering multiplicity of system views and diagrams (9 different diagram kinds in SysML, 48 views in NAF 3.0, 52 views in DODAF 2.0! Can't wait for the third edition…), but most often fail to bring forward a holistic, global, generative view that sums up the intention into one big idea or comprehensive concept. In other words, system engineers are not trained or used to 'pitch' their system in one or few sentences, making us understand why 'we will love it', and how our work may contribute to the value overall.

4 Conclusion and Acknowledgment

It was 'off-the-record', unofficial work performed by the authors, who enjoyed themselves exploring these side topics, and trying to renew our vision of system engineering by exploring development practices in other industries. The analogy with

movie production proved to be fruitful and illustrative, and convinced us to push forward these early forays. Or, as goes the saying, with a rumble voice: We will be back!

We would like to warmly thank Michel Guillerm, a French independent movie director, for the oh-so fascinating discussions we've had with him, and the passionate patience he had leading us through the inner workings of movie making.

5 References

1. The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.2. Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed 20/02/2014. www.sebokwiki.org/
2. Chaos report, Standish Group, 1994..2012, http://blog.standishgroup.com/
3. Vitruve, De Architectura, Vitruve, ~-25 before J.C.
4. System Engineering Handbook, A Guide for System Life Cycle Processes and activities, International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2, version 3.2, January 2010.
5. The Genesis of 'Lost'. The DVD Group, Buena Vista Home Entertainment (2005)
6. Before They Were Lost: Personal Stories and Audition Tapes. The DVD Group, Buena Vista Home Entertainment (2005)
7. Battlestar Galactica making of, Universal Studios, 2008/2009
8. Within a minute: The making of Episode III, Lucas Films, 2005
9. Filmmaking Encyclopedia, Accessed 20/02/2014, http://www.filmbay.com/
10. Filmmaking, Accessed 20/02/2014, http://www.dmoz.org/Arts/Movies/Filmmaking
11. Film Training Manitoba, Accessed 20/02/2014, http://www.filmtraining.mb.ca/
12. Goranson H.T., The Agile Virtual Enterprise: case, metrics, tools. Quorum Books, 1999
13. Watzlawick, Paul. Ultrasolutions. How to Fail Most Successfully. W. W. Norton and Company, Inc. (1988)
14. Wyatt, Justin. High Concept, Movies and Marketing in Hollywood. University of Texas Press, 1994.
15. Frederick, Matthew. 101 Things I Learned in Architecture School. MIT Press, 2007.

Using Orientor Theory for Coherent Decision Making for Application Landscape Design

Alexander W. Schneider and Florian Matthes

Technische Universitt München
Munich, Germany
Email: {schneial|matthes}@in.tum.de

Abstract. More than 30 years have past since the first enterprise architecture (EA) management framework has been published. While the field has reached a certain maturity in science and practice, a paradigm shift is just beginning. The emerging trend to regard an enterprise as a complex adaptive system might allow to redefine and achieve the holistic nature of EA management approaches. Thereby, the focus on static aspects of the architecture might be extended by taking behavioral aspects into account. In this paper, we argue (1) that currently available EA management approaches fall short in achieving the desired holism, i.e. viewing the system (application landscape) as a whole, (2) apply orientor theory to support decision making by deriving coordinated goals and principles for application landscape design and (3) show how a system-of-systems approach of applied orientor theory could look like. We conclude the paper by sketching out an appropriate research agenda.

Keywords: enterprise architecture, complexity, environment, orientor theory, decision support

1 Introduction

For quite some time, researchers regard enterprises as open dynamic systems [34]. Thereby, they overcome the self-centering view of traditional economics and organizational theory as well as the reductionism prevalent in recent enterprise architecture research. Thereby, they also extend their view to take the environment and respective interactions into account [18]. It seems that this extended viewpoint allows for a deeper understanding and development of new methods in the context of accelerating changes requiring steady adaption. Disruptive technologies, increasing regulation and changing customer demands are just a few general examples. It is obvious that enterprise architects have to design the enterprise and especially the application landscape to optimize fitness with respect to these challenges. Therefore, science has to provide methods to support this process. As a first step into this direction, we propose to search for other disciplines which already established similar thinking to develop solutions for similar problems. In this paper we describe one of those modeling approaches which has been successfully used in ecology and economics, namely orientor theory. We

show how orientors might be used to create a formal and deeper understanding of the behavior of application landscapes as a whole taking also their environment into account. Thereby, we not only provide a conceptual integration of different aspects of relevant behavior but also show their dichotomies. Grounded on the assumption that agents with deeper understanding of the system behavior make better decisions the application of orientor theory is a promising tools to model the behavior of enterprise architectures. Finally, we conclude the paper by describing a conceivable road-map for enterprise architecture research that accounts also for dynamics of the enterprise.

2 Problem Statement

2.1 General Goals of EA Management and its Claim for Holism

Even a short literature review reveals that holism is frequently attributed to EA management approaches. A search for the terms "enterprise architecture" AND "holistic" in Google Scholar performed in April 2014 resulted in more than 4.670 articles and book chapters. It is more often than not the case that holism is considered to be a mandatory attribute of EA approaches [25]. Therefore, there is general agreement on the scope of EA initiatives and models. Nevertheless, this is not the case for concrete goals EA management initiatives should pursue. In literature, EA benefits such as improved change management and improved risk management are mentioned frequently [25], some authors also promise an increase of market value [32], better customer orientation and improved alignment with business partners [16]. It becomes obvious that EA initiatives have a wide-ranging scope covering the whole enterprise as well as the enterprise as a whole in its environment.

2.2 Reductionism and Complex Systems

The fundamental idea behind modeling static non-living aspects of an organization with entities and attributes, i.e. EA documentation, to understand or design the system is called reductionism. This philosophical position holds that a system can be completely explained and understood by looking at its constitutive elements and their relationships. Thereby, each phenomenon can be explained in terms of relations between other more fundamental phenomena. In contrast, the science of complex systems teaches us that one inherent characteristic of complex systems is their emergent behavior [14] which is also applicable to organizations [23]. Nowadays, emergent phenomena are considered to be the exact opposite of reductionism, they can hardly be traced back to the interaction of phenomena of single elements. Colloquially, this notion is formulated as "the whole is more than the sum of its parts" which dates back to Aristotle. The problem that arises from the prevalent reductionistic approach of EA management is that the focus on the micro-level is not appropriate to completely explain or even design the behavior on the macro-level. This is especially surprising since, as shown before, the macro-level is the area of interest of holistic

EA management. Furthermore, many researches have shown that organizations qualify to be regarded as complex adaptive systems [34, 12] since they exhibit complex, adaptive and emergent behaviors due to multiple interacting agents [8]. To qualify for a reductionistic approach we need to be aware of the laws declaring how theories on the micro-level, e.g. the resistance to change of a single application, relates to theories on the macro-level, e.g. the resistance to change of an application landscape. As long as science has not accomplished to reveal such laws, a pure reductionistic approach, e.g. by collecting a huge amount of data on the micro-level, is doomed to fail when trying to support design decisions on the macro-level. Although reductionism in general faces criticism in scientific literature (e.g. [15, 20]), we only want to point out here that if reductionism is applied it has to be done on a strong theoretical basis.

2.3 Reductionism is Prevalent in EA Management

A well-known problem in EA modeling is to define the breadth and depth of EA documentations, i.e, relevant elements and their level of detail. If not done correctly issues related to overmodeling and overuse of detail might occur [1]. Typical issues include analysis paralysis, delayed delivery of results and high costs for data collection. The observation that many companies in practice faced such problems clearly indicates that the holistic idea of EA management has often been interpreted by reductionists. Hereby, the assumption seems to be that the more elements and the more details are modeled or documented the closer one gets to a 'holistic' approach. Another way of interpreting holistic in this context would be to look at the whole system, e.g. an application landscape, instead of focusing on its constituting elements.

As mentioned before, (IT) cost reduction is one of the major claims of EA management. Existing literature suggests that EA facilitates building a more standardized IT platform with fewer technologies, leading in turn to simplified interfaces, higher reliability through reduced operating platform complexity, and lower maintenance and support costs [37]. Thereby, the assumed law seems to be that an application landscape's cost is just the sum of the costs of its elements. While this holds true in a narrow context, it might not be true from a holistic point of view. On the one hand, while infrastructure operating costs might decrease, application development costs can increase due to necessary workarounds required in a fixed technology context. On the other hand, cost reductions based on technology standardization and structural complexity reduction are only short term cost optimizations. Risks associated to extensive standardization might cause expenses in the future and have therefore be regarded in respective calculations. To our knowledge, most cost cutting approaches based on EA management neglect the system's of path-dependence. The statically viewed EA has a history which has to be taken into account. For example, investments have been made for some elements in the past while their expected benefits will materialize in the future. Consequently, they cannot be changed without loosing (parts) of the promised benefits. Furthermore, IT systems are not autonomous elements within the application landscape which can

be changed easily. For example, an organization employs IT staff with specific knowledge which could get lost in case of standardization. Additionally, standardization activities on the application landscape might require standardization activities within the business layer.

2.4 A Paradigm Shift in Enterprise Architecting

Despite the fact that many existing EA approaches applied a naive reductionistic thinking even in the absence of concrete laws to build reductions, we see evidence for a paradigm shift in enterprise architecting from reductionism to holistic thinking. For example, the co-evolution path model describes how an enterprise as a whole behaves in the context of its environment [18]. While the complexity of the environment increases, the enterprise has to increase its complexity as well. But an overshot could be very dangerous and therefore each enterprise has to maintain an adequate level of complexity over time. An overview of complexity work in EA management can be found in [33]. Additionally to hard facts as provided by existing EA approaches, we also see an opportunity in supporting EA decisions indirectly via creating a better understanding of the system as a whole among EA decision makers. The benefits of shared mental models, conventions and language have already been recognized in the context of virtual teams [22], customer relationship management [27] and lab experiments in general [35]. Therefore, we argue that models facilitating such shared mental models or language should also be established for the domain of enterprise architecting because a shared IT-business understanding allows companies to conceive, implement and use innovative IT applications to improve process performance [28].

3 Applying Orientor Theory to Model Application Landscapes in their Environment

As outlined before, we want to extend the scope of EA models towards achieving a holistic view. A literature review conducted in 2014 revealed, that currently the dynamic aspect as well as the subjective aspect of an enterprise's complexity is underrepresented in EA literature [33]. Therefore, we employ orientor theory as a method which might be able to create such shared mental models in the context of application landscape design. Bossel [5] defines orientors as the "set of criteria that are relevant for the evaluation of system development [...] that systems (or their managers) use to orient their decisions and actions regarding the system". Although orientor theory originates from ecology, it has been used to describe complex systems in arbitrary contexts [6].

3.1 Benefits of Orientor Theory for Application Landscape Decisions

The application of orientors is a means to cope with situations in which the desired state of a system is not agreed upon the designers. Due to the large

number of different interest groups directly involved in planning and decision-making processes influencing an application landscape, several points of view have to be taken into account. Especially in ecosystems, where orientor theory has been developed, in such situations a single objectively derived best solution does not generally exist [13]. Therefore, orientors form conditions that we can apply to systems in order to judge their sustainability [36]. Because decisions about extensive application landscape (AL) transformations are made in boards where people with different views and goals converge, the application of orientors and especially their dichotomy can help to establish shared understanding of the problem among the decision makers. That shared problem understanding can help establish development priorities and keep senior management focused on generating benefits from new IT capabilities has already be shown in practice [31]. In addition to that, orientors also offer strategic guidance for decision making processes on different levels of governance. A framework based on orientors would allow to balance different interest and therefore ensure the whole system's viability.

Furthermore, the application and modeling of orientors not only allows to assess the current state of a system and better understand opposing goals but also allows to agree upon desired orientors the system should follow. Based on such framework and concrete strategies goal derivation can be facilitated. Therefore, orientor theory should be of interest especially for EA approaches following the Enterprise Ecological Adaptation school of thought [19], whereby sustainability and organizational coherence are major goals.

3.2 Orientor Theory and its Implications for EA Management

According to orientor theory each system orients towards six environmental aspects namely normal state, scarce resources, variety, evolution, variance and other systems. Here, we focus on the AL as the system under investigation including applications, infrastructure as well as the people interacting with them. Figure 1 depicts the six pairwise contradictory orientors and their respective environmental influences.

Existence The existence orientor refers to the normal state of the environment which means that the system has to maintain its state variables constant to enable functioning under given circumstances. This orientor requires:

- A protective shell preserving the systems from threats able to push the system state out of the acceptable range
- No failure of system structure
- No self-destructive behavior

Following the existence orientor an AL would, e.g., install firewalls to protect applications from hackers. To prevent failures of the system structure all relevant stakeholders have to be involved in AL design decisions which in practice is still an issue [21]. IT capabilities for this orientor include IT service management as well as a monitoring capability.

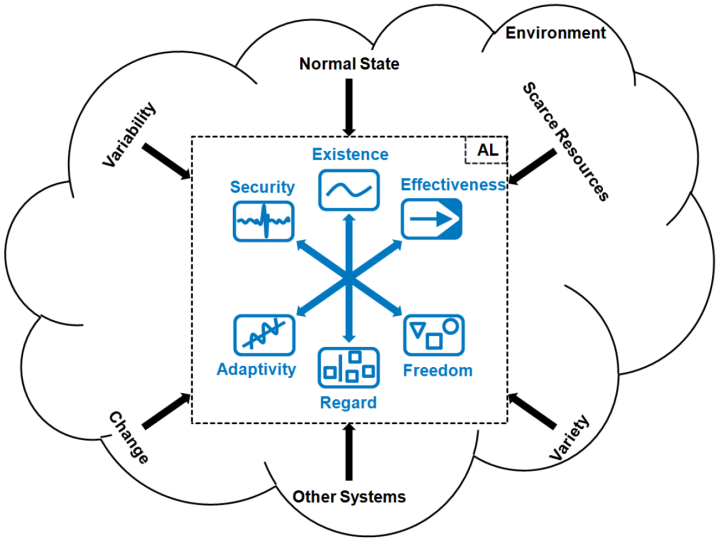


Fig. 1. Application landscapes modeled according to orientor theory [5, 10]

Effectiveness The effectiveness orientor refers to scarce resources in the environment and how they can be secured. Within the system, resources have to be distributed wisely and used in an efficient way, e.g. budget, time and knowledge. Externally, the system has to ensure efficient acquisition of scarce resources by connections to the environment and other subsystems. Knowledge acquisition through hiring new employees can be difficult, for example in the domain of EA management [21]. In order to acquire budget, executive support or management commitment is needed. But this is often rather low regarding EA management which is often seen as an operational initiative rather than a strategic concept with long-term redemption [17].

Following the effectiveness orientor an AL has to continuously balance efforts to make processes and applications more efficient with efforts creating assets which are non-efficient but effective in the long run. The better the ratio of efforts and outcomes the more orientation towards effectiveness. But that does not imply that the ratio has to be good at any point in time. To stay viable the system has to maintain a positive ratio on average over time. Therefore, an integration of EA management and project portfolio management is inevitable [11]. However, the system needs to ensure access to required resources from the environment. For application landscapes this implies, e.g., access to people skilled in programming languages in use or skilled enterprise architects which are still issues in practice [26, 21]. Another means to accomplish efficiency is to increase standardization within the AL or especially within the infrastructure layer.

Freedom The freedom orientor refers to the variety of the environment and describes the system's freedom of action. Thereby, a system needs as much freedom as its environment offers variety [2]. In general, a system has to secure itself from overextension by using one of the following strategies:

- Reacting by using the systems repertoire
- Reacting by influencing the system's environment
- Reacting by searching for a new environment

Following the freedom orientor an AL would try to achieve a maximum of action alternatives and limit the amount of fixed structures. These include but are not limited to long-term contracts with infrastructure or application providers, a high penetration of a single vendor within the AL as well as non-redundant processes and applications. Another ability relevant for ALs is to cope with technological progress, especially with disruptive technologies as they might limit the whole system's viability if the system is constrained in its action alternatives. Some companies even changed their environment in presence of a significant environmental, i.e. external, change by moving their headquarter overseas [3]. Employing diverse workforce, having a modular AL and decentralized governance structures also supports the freedom orientor.

Adaptability The adaptability orientor regards the evolution of the environment. If a system is not able to elude from threatening influences it has to adjust its parameters or even its structure. In general, systems can either adapt their structure or their behavior. Changing the system's structure can result in a new system differing explicitly from the old one. In contrast, changing only the behavior is also considered as co-evolution and which is mostly suitable if small environmental changes occur. Such adaptability requires a certain degree of self-organization. In particular, the following conditions facilitate adaptivity:

- versatile system components
- variety within the system structure
- redundant but physically different processes
- decentrality and partial autonomy
- memory as information storage to enable learning

Following the adaptivity orientor would require, e.g., a modular architecture with loosely coupled applications, data and technology components which allows to set global standards while also allowing regional differences [30]. In order to apply versatile components and variety within the AL applications can be build on different technologies and programming languages and conscious acceptance of functional redundancy. Since adaptivity requires the ability to learn knowledge management becomes vital for adaptive ALs, cf. [9]. An example for a structural change of the AL is a transition to a service oriented architecture. Fostering an open organizational culture and having flexible structures directly supports the adaptivity orientor.

Security The security orientor regards temporal variances of influences and ensures that the system is safe from unforeseen harmful influences. Therefore, the system needs to be mostly independent from unstable environmental factors and dependent only on stable environmental factors. In general, this can be accomplished, e.g. by

- setting up buffers to contain overloads and bypass supply gaps
- establishing self-regulating structures
- defusing potentially harmful threats

For application landscapes following the security orientor would imply, e.g., to keep enough knowledge within the company to overcome potential supply gaps on the market. Furthermore, to establish self-regulating structures an comprehensive monitoring and governance capability is needed. Defusing harmful threats in this context could be achieved, e.g., by setting up uninterrupted power supply units or different connections to the Internet. Setting up resource buffers, e.g. via server virtualization, secures individual applications from request overloads. Especially the implementation of appropriate risk management and continuity management support the security orientor.

Coexistence The coexistence orientor refers to other systems influencing the system and anticipative behavior. Each system has to consider the behavior and interests of other systems for its own interest. Usually, each influence from the environment has an 'unsystemic' component as well as a component consisting of the behavior of other systems. Since sometimes a specific system is of special interest, that system has a special role within the coexistence orientor. It requires:

- the ability to realize that another system is affected by some influence
- the availability of behavioral patterns

While one can imagine a huge amount of relevant systems for an application landscape an important one should be the enterprise or business units. Being informed about current strategies, e.g. to increase the number of customers for a certain product, allows the application landscape to invest in respectively required capabilities such as scalability. Other systems of interest could be providers of infrastructure or applications. Environmental influences like the dissemination of cloud computing might influence the adaptivity of them and therefore indirectly influence the AL. Therefore, sensors as well as analytics capabilities are required for the coexistence orientor.

3.3 Mapping Established Enterprise Architecture Principles to Orientors

In order to steer an enterprise architecture (EA) in general and an application landscape (AL) in particular adopting EA principles is a pervasive means, cf. [29]. Because principles are used to steer an application landscape towards a specific

direction, they obviously qualify to be mapped on the six basic orientors. We will do this for three exemplary EA principles formulated in the industry standard TOGAF (The Open Group Architecture Framework) [38].

Common Use Applications "Development of applications used across the enterprise is preferred over the development of similar or duplicate applications which are only provided to a particular organization". This principles directly orients towards effectiveness because the intention is to save costs and time during implementation and operation of different IT systems performing the same tasks. Thereby, it renounces the freedom as well as the adaptivity orientor. If applied, freedom in term of action alternatives will be limited because, e.g., the number of available add-ons is limited if only one solution is used. Setting global standards also prevents or impedes local evolution and therefore limits the adaptivity of the AL.

IT Responsibility "The IT organization is responsible for owning and implementing IT processes and infrastructure that enable solutions to meet user-defined requirements for functionality, service levels, cost, and delivery timing". This principle obviously orients towards the existence of the AL because defined responsibilities ensure that someone really cares about a system or entity. However, the principle reduces the regard to other systems, e.g. the business units, because keeping the AL viable might become more important than keeping the depending business units viable.

Technology Independence "Applications are independent of specific technology choices and therefore can operate on a variety of technology platforms". This principle clearly orients towards adaptivity because if applied it allows for a grater variety of components. On the one hand, e.g., if a new data storage providing better performance is offered by the environment the AL can exploit these performance gains. On the other hand, the orientation towards effectiveness is lowered because defining and managing interfaces and shared protocols which might be subject to evolution increases costs.

3.4 A System of Systems Approach

In the previous section we outlined how orientor theory could be used to model the role of application landscapes (AL) within their environment. But since ALs are designed systems such orientor model could also be used to support design decisions. Therefore, an enterprise architect has to decide which orientor is most important for the AL. But, an EA or AL could also be regarded as a system of systems [24] wherein the behavior of each individual is explained by the structure and arrangement of the lower individuals of which it is composed [7]. Because in such setting each sub-system again can be regarded as a system and modeled with orientors we can use the orientor approach to model ALs,

domain landscapes, application clusters as well as single applications. Although such approach has already been proposed in general [4], we want to elaborate the relationship of systems' and sub-systems' sustainability here. Until now, the system-of-systems orientor approach assumed that the system's sustainability depends on the sustainability of each sub-system. The challenge is to identify the different orientors responsible for the system's sustainability, i.e. viability of the company. In the context of ALs, this could mean that the sub-system consisting of all applications supporting production processes have to orient more towards effectiveness and therefore standardize protocols and vendors whereas the sub-system consisting of all customer-facing applications has to orient more towards adaptivity since the environment is rapidly changing, e.g. mobile devices. But we also want to point out, that the system's viability can also be achieved by explicitly leaving sub-systems to die. This includes, e.g., IT carve-outs or discontinuance of a whole line of business.

4 Conclusion and Outlook

In this paper, we argued why existing EA management approaches fall short in providing a holistic approach and introduced orientor theory as a means to describe a system, i.e. application landscape, as a whole in the context of its environment. By linking aspects of existing EA management approaches to respective orientors we put them into a more general and coherent framework and thereby identified opposing forces. Furthermore, for each of the six orientors we derived implications for AL management and outlined how a system of systems approach could be used to apply orientor theory for sub-systems like domain landscapes or application clusters as well. In addition, we mapped well-known EA principles to the six basic orientors, identified that their application shifts an AL towards one orientor while dismissing another and thereby demonstrated the benefits of applying orientors for AL management. In order to underpin the applicability of the proposed modeling method researchers have to observe its application in practice. If the framework should be used to assess and steer application landscapes concrete measures have to be defined for each orientor. In case of a sufficient data base we suggest to analyze, for example, if companies within the same industry branch or of equal size orient their application landscape development towards the same orientors. It would also be worthwhile to examine the use of domain-specific orientors empirically. Furthermore, we suggest to analyze other approaches which are able to model system behavior, such as causal loop diagrams, in the context of application landscape management in order to model behavior in more detail.

References

1. Armour, F.J., Kaisler, S.H., Liu, S.Y.: Building an Enterprise Architecture Step by Step. IT professional 1(4), 31–39 (1999)
2. Ashby, W.R.: An introduction to cybernetics. Methuen, London (1976)

3. Birkinshaw, J., Braunerhjelm, P., Holm, U., Terjesen, S.: Why do some multinational corporations relocate their headquarters overseas? Strategic Management Journal 27(7), 681–700 (2006)
4. Bossel, H.: Assessing viability and sustainability: a systems-based approach for deriving comprehensive indicator sets. Integrated Natural Resource Management: Linking Productivity, the Environment and Development pp. 247–266 (2003)
5. Bossel, H.: Systeme, Dynamik, Simulation: Modellbildung, Analyse und Simulation komplexer Systeme. Books on Demand, Norderstedt (2004)
6. Bossel, H.: Systemzoo, Systemzoo, vol. 3. Books on Demand, Norderstedt (2004)
7. Boulding, K.E.: General systems theory–the skeleton of science. Management science 2(3), 197–208 (1956)
8. Brown, S.L., Eisenhardt, K.M.: The Art of Continuous Change: Linking Complexity Theory and Time-Paced Evolution in Relentlessly Shifting Organizations. Administrative science quarterly 42(1), 1–34 (1997)
9. Buckl, S., Matthes, F., Schweda, C.M.: Future research topics in enterprise architecture management–a knowledge management perspective. Journal of Enterprise Architecture 6(3), 16–27 (2010)
10. Burger, M.: Modeling Application Landscapes as Dynamic Systems. Master Thesis, Technische Universität München (2013)
11. Dern, G.: Management von IT-Architekturen: Leitlinien für die Ausrichtung, Planung und Gestaltung von Informationssystemen. Praxis, Vieweg + Teubner, Wiesbaden, 3., durchges. aufl edn. (2009)
12. Fuller, T., Moran, P.: Small Enterprises as Complex Adaptive Systems: a Methodological Question? Entrepreneurship & Regional Development 13(1), 47–63 (2001)
13. Gnauck, A.: Applying Ecological Goal Functions: Tools for Orientor Optimization as a Basis Decision Making Processes. In: Müller, F., Leupelt, M. (eds.) Eco targets, goal functions, and orientors, pp. 511–525. Springer (1998)
14. Goldstein, J.: Emergence as a construct: History and issues. Emergence 1(1), 49–72 (1999)
15. Horst, S.W.: Beyond reduction: Philosophy of mind and post-reductionist philosophy of science. Philosophy of mind, Oxford University Press, Oxford (2007)
16. Jonkers, H., Lankhorst, M.M., ter Doest, Hugo WL, Arbab, F., Bosma, H., Wieringa, R.J.: Enterprise architecture: Management tool and blueprint for the organisation. Information Systems Frontiers 8(2), 63–66 (2006)
17. Kaisler, S.H., Armour, F., Valivullah, M.: Enterprise Architecting: Critical Problems. In: 38th Annual Hawaii International Conference on System Sciences (2005)
18. Kandjani, H., Bernus, P., Nielsen, S.: Enterprise Architecture Cybernetics and the Edge of Chaos: Sustaining Enterprises as Complex Systems in Complex Business Environments. In: 46th Hawaii International Conference on System Sciences (HICSS). pp. 3858–3867 (2013)
19. Lapalme, J.: Three schools of thought on enterprise architecture. IT professional 14(6), 37–43 (2012)
20. Lewontin, R.C.: The triple helix: Gene, organism, and environment. Harvard University Press (2001)
21. Lucke, C., Krell, S., Lechner, U.: Critical Issues in Enterprise Architecting – A Literature Review. In: Proceedings of the 16th Americas Conference on Information Systems (AMCIS) (2010)
22. Maynard, M.T., Gilson, L.L.: The Role of Shared Mental Model Development in Understanding Virtual Team Effectiveness. Group & Organization Management 39(1), 3–32 (2014)

23. Morel, B., Ramanujam, R.: Through the looking glass of complexity: The dynamics of organizations as adaptive and evolving systems. Organization Science 10(3), 278–293 (1999)
24. Morganwalp, J., Sage, A.P.: A system of systems focused enterprise architecture framework and an associated architecture development process. Information, Knowledge, Systems Management 3(2), 87–105 (2003)
25. Niemi, E.: Enterprise architecture benefits: Perceptions from literature and practice. In: Niemi, E., Ylimäki, T., Hämäläinen, N. (eds.) Evaluation of enterprise and software architectures: critical issues, metrics and practices. University of Jyväskylä (2008)
26. Paulson, L.D.: IT hiring growth modest, but steady. IT Professional 8(1), 6–9 (2006)
27. Plouffe, C.R., Williams, B.C., Leigh, T.W.: Who's on First? Stakeholder Differences in Customer Relationship Management and the Elusive Notion of Shared Understanding. Journal of Personal Selling and Sales Management 24(4), 323–338 (2004)
28. Ray, G., Muhanna, W.A., Barney, J.B.: Competing with IT: The Role of Shared IT-Business Understanding. Communications of the ACM 50(12), 87–91 (2007)
29. Richardson, G.L., Jackson, B.M., Dickson, G.W.: A principles-based enterprise architecture: Lessons from Texaco and Star Enterprise. MIS Quarterly pp. 385–403 (1990)
30. Ross, J.W.: Creating a Strategic IT Architecture Competency: Learning in Stages. MIS Quarterly Executive 2(1), 31–43 (2003)
31. Ross, J.W.: Enterprise architecture: driving business benefits from IT. Massachusetts: Massachusetts Institutue of Technology (2006)
32. Schekkerman, J.: The economic benefits of enterprise architecture: how to quantify and manage the economic value of enterprise architecture. Trafford Publishing (2005)
33. Schneider, A.W., Zec, M., Matthes, F.: Adopting Notions of Complexity for Enterprise Architecture Management. In: Proceedings of the 20th Americas Conference on Information Systems (AMCIS) (2014)
34. Schneider, M., Somers, M.: Organizations as Complex Adaptive Systems: Implications of Complexity Theory for Leadership Research. The Leadership Quarterly 17(4), 351–365 (2006)
35. Selten, R., Warglien, M.: The Emergence of Simple Languages in an Experimental Coordination Game. Proceedings of the National Academy of Sciences 104(18), 7361–7366 (2007)
36. Spangenberg, J.H.: Biodiversity pressure and the driving forces behind. Ecological Economics 61(1), 146–158 (2007)
37. Tamm, T., Seddon, P.B., Shanks, G., Reynolds, P.: How Does Enterprise Architecture Add Value to Organisations? Communications of the Association for Information Systems 28, 141–168 (2011)
38. The Open Group: TOGAF Version 9.1 (2011), `http://pubs.opengroup.org/architecture/togaf9-doc/arch/`

Reuse / Variability Management and System Engineering

Olivier Renault
PLM Industry Expert
PLMSys Consulting
111, avenue Victor Hugo – 75784 Paris Cedex 16
olivier.renault.pro@gmail.com

Abstract: this paper aims to share industry experience in managing the reuse and variability in the industry and to analyze the linkage between this topic and system engineering. Reuse and variability management are two faces of the same coin and strongly influence business performance. Hard products industries (where mechanical and structural engineering were historically dominant) and soft systems industries (where electronic and software engineering are dominant) addressed the questions from different perspectives. After describing the observed practices for managing the reuse and variability from the physical product standpoint, and taking in account concepts and approaches used in "Soft" industries, we analyze how systemic approach should help in better mastering the variability. In conclusion, we identify some principles and rules which would need to be investigated through research to better link PLM and systemic approach in variability management

Keywords
Variety, Variability, Reuse, Options, Variants, Configuration, Product Families, Product Lines, Requirements, Features, Systems, Platforms, Modules, Architecture, Interfaces, Product Life Cycle, PLM Systems

1 Introduction

As we will see across this paper, variability and reuse management is a topic which crosses the full life cycle of complex products and systems and address all the engineering disciplines. One of the identified difficulties is that variability management have often been addressed separately by different methods and IT systems according to their positioning on life cycle and engineering disciplines. My own experience presented in the first sections of this paper, is mostly relevant to the management of the variability implemented in PLM and ERP systems. It focuses on variability management relying on product structure (from the classical perspective of BOM management supported by both classes of systems, but also from the perspective of the 3D Digital Mockups used by a large bench of

engineering disciplines for concurrent engineering. But this perspective is too limited. I intuitively thought that system engineering should be analyzed in the variability management context for two main reasons:

- First, electronics and software is massively invading all traditional products whose design and development previously relied on structural and mechanical design. Even if these engineering disciplines are often treated with separated methods and dedicated IT application, mutual dependencies are growing. Complex systems/product definition need to be managed from a consistent point of view for configuration, including variability management and changes across time.
- Secondly, my experience focused on physical product variability management and the problems encountered convinced me that a systemic approach is required to better manage the variability. This relies on the fact that variability is strongly linked to the way we structure and manage the requirements in parallel with the definition of the system/product architecture.

This also pushed me to do a brief research review where I found interesting papers. A part of them were centered on the domains of the physical product variability. But I also discovered that a lot of the research papers address the question of variability from the software engineering perspective. These papers helped me to clarify and confirm some intuitions I got from my own experiences. They also help me to formulate the reasons why I think that system engineering and system modeling approaches may provide a strong foundation to design and model the variability. They provide a way to consistently articulate variability and architecture definition during the development process of complex systems /products.

2 Business approaches and drivers for variability management in industry.

We will mention here two industries which showed a strong concern on variability management, just to underline the business impact it had.

In the Information Technology industry, IBM introduced early in the 60s an ambitious modular and configurable architecture of business computers with the IBM 360 systems. This program was very successful and constituted one of the main reason of the further dominance of the market by IBM [Manet Hamm O.Brien 2011]). IT industry is now an industry where standards (OS, telecoms, DB, Internet….) and layered architecture took an essential part in its uninterrupted growth.

The automotive industry is probably one, in hard products domain, which faces among the most complex challenge to manage variability. A car is a technical complex object (several thousands of parts) and has to sustain a very large variability [Jiao, Simpson, & Siddique 2007]. [Volkswagen 2011] & [Renault Nissan 2013] summarize the respective approaches of Volkswagen and Renault-Nissan groups to manage their variability based on approaches to platforms and modules. We may summarize them by the following principles.

- Vehicles of these brands are organized by vehicle families. Often a vehicle family covers one market segment for a brand and includes all types of bodies for the brand and the segment, with different possible engines. One vehicle family generally reuse one single platform family (sometimes two for market–cost optimization).
- A platform family is generally common to several vehicle families of one or several brands and may cover one or more vehicle segments. The platform integrates the chassis and all equipment generally hidden to the customer while the complementary part of the vehicle includes all elements of style directly perceived by the customer.
- Common modules are designed to equip the whole range of platform families in order to maximize the scale economy. A similar function (e.g. a seat), may be implemented through one, or a very limited number of module families. Each module generally include the variability required to cover common market needs for all platform and vehicle families.

Multiple sources of research papers and articles develop the business drivers for variability and reuse management.

- [Jiao, Simpson, Siddique 2007] provides a comprehensive review of state of arts research on product family design and platform-based products As part of the work the economic justification section references most important papers on this topic..
- For the software industry, an economic model is proposed by [Rokunuzzaman & Choudhury 2006]. It estimates benefits to reuse software components for building a customized software solution. [Lim 1994] gives metrics collected during two reuse programs of Hewlett Packard.
- [Pil, Holweg 2004] analyzes the variability and its economic drivers focusing the study mainly on the automotive sector. They gives order of magnitude of the variability and focus on how the products variability has to be linked to the order fulfillments strategy. Their paper well illustrates the strong focus that hard products industries had to manage variability from the physical product and supply chain perspective.

Table 1 hereafter proposes a summary the main business drivers for the variability and reuse management for hard products industries.

Table 1

Drivers for Variability	**Drivers for Reuse**
(B2C) Customer diversity of demand (ie: Car bodies, painting, engine, equipment's …)	Supply chain costs and delays reductions • Production (scale effects) – Internal / External (supply chain) • Standardization / flexibility of production process and facilities • Capacity planning • Order to delivery delay • Quality improvement (repeatability) • Development reduction cost (initial and change management)
(B2B) Ordering company differentiation (ie: Aircraft companies specific cabin layout – engine – electronic systems variants, length, capacity, mission…)	Development delays / costs / risks • Development reduction cost (initial and change management) • Innovation value focus • Risk minimization • Tests validation reduction
(B2C – B2B) Country constraints • Regulations • Climate • Customer Usages • Infrastructures …	Innovation • More focus on value innovation • Faster introduction in existing products (standardized modular architectures)
(B2C – B2B) New Technologies introduction	Flexibility – Speediness to change and adaptation. (same changes to apply on a wider spectrum)

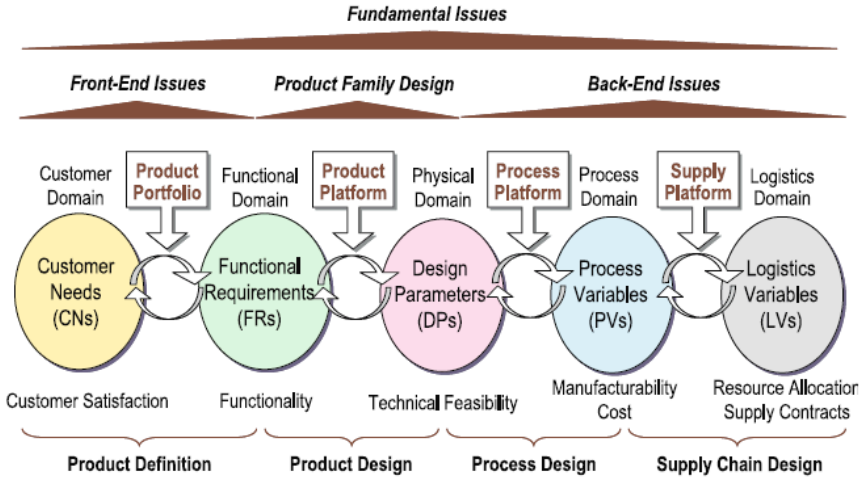
[Main business drivers for variability and reuse for hard products industries]

3 Approaches and gaps for managing variability in IT Systems.

3.1 Management of variability in CRM, MRP/ERP and PLM

As explained by [Pil, Holweg 2004], the variability management encompasses the full life cycle of the products. This is illustrated by Figure 1 hereunder reproduced from their paper.

Figure 1: [Pil, Holweg 2004] – Holistic view of Product Family Design and development.



CRM (Customer Relationship Management), MRP/ERP (Manufacturing Requirements Planning/Enterprise Resource Planning) and MRO (Maintenance, Repair and Operations), each of these systems manage the product during one of its "physical" life cycle stages. So they are impacted by reuse and variability management. Variability management requires capabilities for each of these domain systems. They may be standard capabilities provided by market software packages, but they also often rely on specific development extending these software package or working as stand-alone systems/applications.

The industry experience I present in this paper is mainly focused on practices used in PLM (Product Life Cycle Management). PLM is the system used to support the design and definition of products and of the life cycle processes and resources related to these products. The PLM supports and coordinates all engineering disciplines and manages all the technical information attached to the product and its product life cycle. So, PLM is placed at the critical stage where variability and reuse are designed. PLM architecture is built around a PDM (Product Data Management) system which offers central services to store, retrieve, classify, and configure all technical data (models and documents). The different applications or systems used to sustain each of the engineering discipline activities are commonly articulated and integrated with the PDM central system to build the overall PLM system architecture. Software engineering and configuration management remains relatively autonomous. Nevertheless there is need to better integrate them within the PDM systems to enable a consistent multi-level configuration management.

PLM variability definition has vocation to be the reference basis for the configuration models used by CRM and MRP/ERP systems or applications. The configuration models of the product families used by each system need to be maintained and synchronized across the change management process.

3.2 Main practices used in PLM for managing the variability

We present here PLM observed practices for managing the reuse and variability. We compare practices mainly applied for managing the reuse and variability in product structures representing the physical product. As explained in section 4, for hard product industries, requirements and functional variability were historically managed through documentation in a traditional development process. Impact of system and software massive intrusion in these industries changed the game. But, from our experience, we still see system and software and PDM managed very separately.

In the following, we will use the words

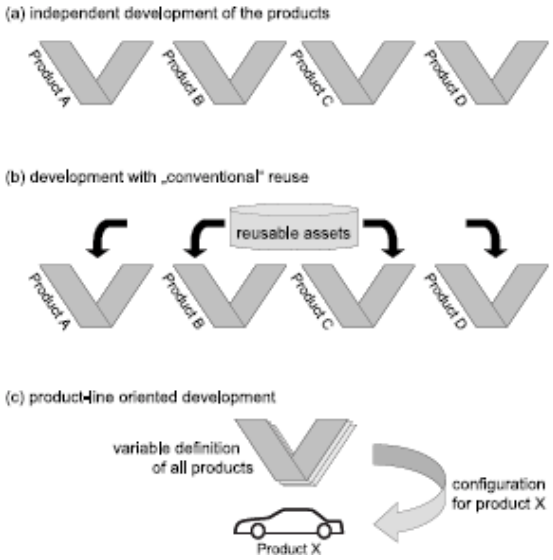
- system/product to specify the high level "final" system/product which has to be designed and developed and which represents the higher level of integration. If, in physical product structure, we should speak only on product, we extend the concept to system/product in the perspective of system integration in our analysis as presented in section 4.
- Sub-systems/modules to specify the element of a product/system definition which could be reused between different higher level systems/products. As the high level product is represented by a product structure defining its composition, a module may be itself defined by a product structure representing its own product composition.

The 3 dominant practices are summarized in Figure 2 extracted from [Reiser 2009]

Practice a: Duplicate and specialize systems/products structures (independent development of products)

The principle is to create a specific system/product structure for each (top level) product. The reuse is done by initial copy or several partial copies from structures of similar system/product. The inconvenience of this approach is that the duplication of common elements encourage the specialization of the definition, even if there is a significant business advantage to maintain a common definition.

Figure 2: Product portfolio development approaches [Reiser 2009]



Practice b: Separated products top structure sharing common configured product components (development with conventional reuse)

In this approach, common sub-systems/modules internal structure are instantiated in each top system/product structure, but remain unique. If the common sub-systems/modules holds variability, it is only the resolved configurations which are instantiated where needed. The main advantage is that this approach forces to maintain a better communality of sub-systems/modules across the different systems/products where they are used. The constraints and limits are:

- sub-systems/modules need to be designed to address requirements of future product-systems (at least at the architectural design level).
- changes to sub-systems/modules need to be controlled with all different upper levels systems/products using them.
- sub-systems/modules variants need to be explicitly configured (specific references) in the upper-level systems/products structure.
- on multi-level architecture, rules for managing and updating the configuration of the different systems/products (number of Configuration Items (CI) levels –revision number absorption levels and rules), and process for propagating changes on upper levels need to be carefully designed to minimize the number of revision updates.
- when there is a large number of combinations of options-variants, impacted by a change, the process to update and maintain all these combinations may be complex. Revision numbers have to be updated on

all parent structure representing these combinations and may need some PLM automation. A good approach is to group several changes and to apply revision number changes on the upper structure only when the group of lower level changes has been fully defined and validated.

- Another condition is to really have a unique structure to maintain the common part of different configured variants structures. On the contrary a change in the common part implies to update each configured variants where it is duplicated. This could be painful and leads to error if this is not automated in some way.

Practice c: Define a common system/product family structure.

This practice consists in creating a unique structure for product family holding the whole variability description for the family. This rely on options – variants mechanisms. This practice is detailed in the following section.

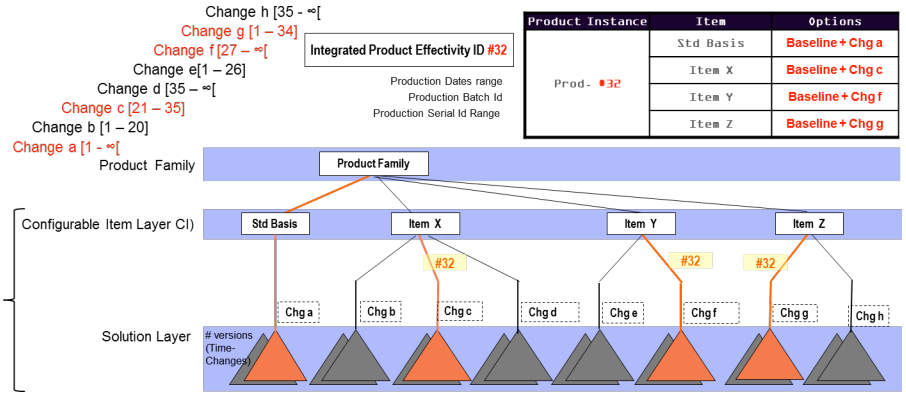
3.3 Practices used in PLM for managing system/product family (product lines oriented development)

Practice a: Product family unique structure carrying the variability description for the whole family by production effectivity

The principle is to have a unique structure for a family of products and to associate a production effectivity to the proper elements of the structure. A production effectivity is generally a set of serial numbers, a range of date delimiting a batch of production for the same products, a batch ID. The configuration of one specific system/product instance (physical product produced or planned to be produced) may be retrieve by selecting all structure items with a product effectivity matching its own production ID. This mechanism may also be implemented through a change management process as summarized in Figure 3 here under.

This approach seems well suited to industries where the variability is driven by a custom to order process where specificities of each variant/option cannot be anticipated and may be very specifically linked to the order requirements for customization.

Figure 3: Product family with variability managed by production effectivity



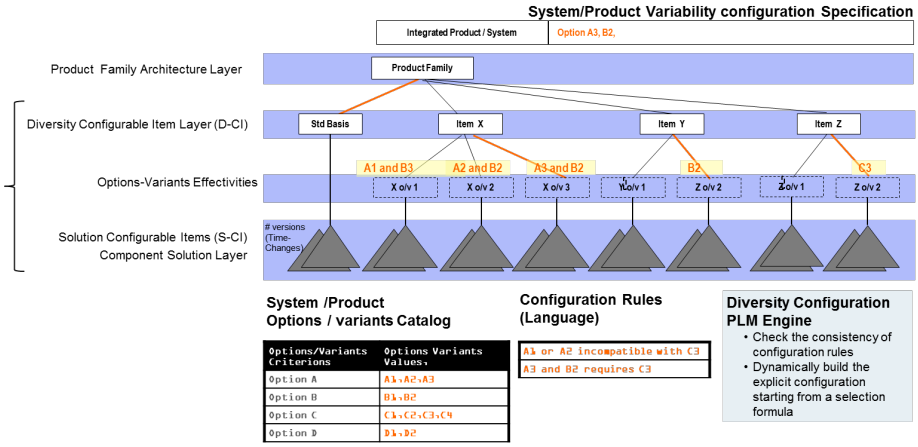
Practice b: Product family unique structure carrying the variability description for the whole family by variability effectivities and logical rules

This principles are the following:

1. Describe the possible option/variants by a language based on:
   a. variability criteria objects holding each one a "dimension of variability"
   b. variability criterion values, each value corresponding to a variability possibility inside the same variability dimension
2. Associate to item nodes in the structure a logical expression corresponding to the combination of option/variant values for which the structure under the item has to be retained. We name it variability effectivity or effectivity. This logical expression may contain logical operator such as NOT, AND, OR…
3. A set of rules may be added to define compatibilities or dependencies of different options/variants.
4. Finally, to define a particular product in the product family, we must select directly or indirectly one value for each option/variant criterion proposed at the family level. The selection request to configure one system/product instance may be explicit. In this case we must express each criterion object with its possible options/variants value (any number). Or it may be implicit (all values of criteria not specified are, by default, retained). The selection is then modified or rejected against the set rules for dependencies and compatibilities.

The Figure 6 summarizes the main principles for describing the variability of the product family in a unique product family structure.

Figure 6: Product family unique structure using based on variability effectivity and rules



When both the number of potential combinations and the volume of production are high, this approach enables us to directly and dynamically "solve" the configuration of the configurable structure by selecting the options/variants value for each criterion. But there are some constraints and limits:

1. The system/product family configurable structure need to be properly defined (system/product items with, for each of them, the proper variability effectivity defining the options/variants for which this item substructure and definition is applicable). This implies the two following rules to be verified.
   - Any selection of options/variants values should not lead to an incomplete configured structure (no function/parts "holes" in the configured structure).
   - Any selection of options/variants values should not lead to a number of system/product items selected beyond the number expected (no function/parts "bump" in the configured structure).
2. If this approach is basically used, the variability of a sub-structure has to be configured by criteria defined for the overall system/product family structure. This could lead to the impossibility of reusing this configurable sub-structure in another system/product family. In the automotive industry, platforms and modules variability management cannot be done in a single system/product family without strongly complicating reuse.

Practice c: Multi-level Product families

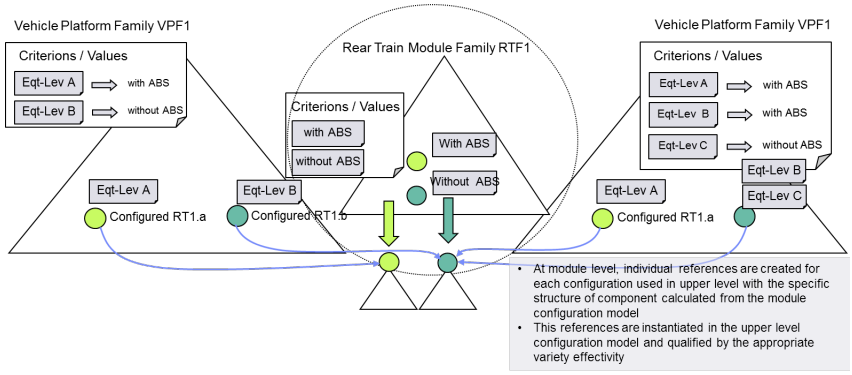
It may be necessary to manage variability at different systems/product structures levels. This is the case in automotive for vehicles, platforms and modules.

In this example, each level (vehicle, platform, sub-systems/modules) holding variability must be considered as a distinct system/product family (product line). The vehicle family structure will instantiate the platform family structure it reuses. Each of them (vehicle or platforms) will also instantiate the sub-system/module families structures they reuse. Applying variability effectivities principles, each family is supposed to have its own variability definition relying on a specific set of variability criteria and criterion values.

When configuring a specific configuration at the higher level (ie: vehicle) by selecting value for each criterion specified in this family, three approaches may be used to select the proper variability of the lower level families.

- Approach 1: Instantiate in the upper-level family structure only configured options variants of lower level family (see Figure 4 hereunder). Each configured option/variant of the lower-level family will be characterized in the upper level structure by a use-case (logical expression of options/variants of the upper family in which this structure has to be configured). The inconvenience of this choice is that the maintenance of the configured lower family structures may be heavy in case of changes. This is especially true when changes are located in their common parts and when the number of configured structures for the lower family is important.

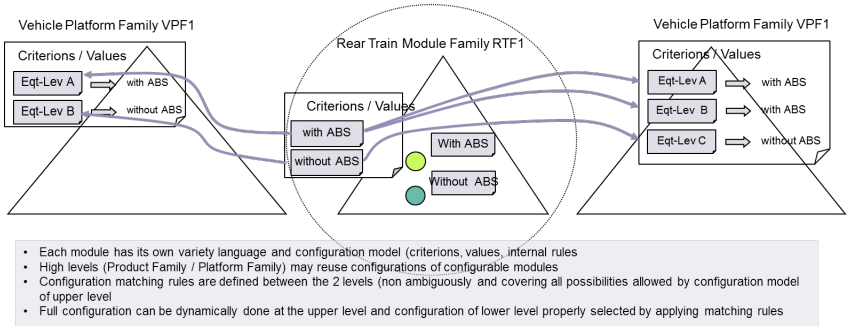
Figure 4: Instantiation of configured structure of the lower level family.



- Approach 2: Reflect all criteria and criterions values of the lower level families in the upper level. For example, all platforms and sub-systems/modules criteria and criterions values will be directly included and visible in the set of criteria used at the vehicle level. We may easily understand that this approach will strongly increase the number of criteria used at the higher level and make the configuration process complex. In other words, with this approach, we expose all the internal variability of the lower family levels to the higher levels even when this does not create value. (e.g.: expose the internal variability of the seat with its options motor and heating when, at the vehicle level we want to use only a couple of criterion such as level of equipment and country/geography of sale to drive the internal configuration of the seat). This approach may be impracticable.

  Approach 3: Define the selected options/variants of the lower level family by rules enabling to convert variability effectivity of the higher level family to variability effectivity of the lower level family. This approach enables us to simplify the number of criterions used at the higher level, while enabling the use of a larger number of options/variants at the lower level. The complexity of the work is to define the mapping rules and to be sure that this mapping enables the effective respect of the two basic rules ("no holes", "no bump") in any resolved configuration. It is summarized in Figure 5 hereunder.

  Figure 5:Multi-level families with variability effectivity mapping



3.3 Variability management in engineering – The question of 3D configurable Digital Mockups

This is a complete topic which needs to be developed more. We will only summarize the main outcomes of our experience here without explaining them in detail.

The principle main goal of the 3D Digital Mockup (DMU) is to integrate all the 3D definitions of the different components in a common model making a complex product. In this way, it sustains concurrent engineering and allows 3D Design in context.

The first concern of variability management in DMU is that each engineering team needs to design the whole variability corresponding to the product items and potential families it has in charge. But reversely, it must take care only of the surrounding variability which could affect it. The approach of 3D Design in Context based on DMU pushes some companies to fully configure these mockups with variability and to do it at part level. Even when achieved, the observed fact is that it is very difficult for engineering teams to select among the multitude of surrounding variability combinations those which are the most constraining for their design. That is why I recommend to model physical architecture of the product in the DMU. For one product family, we may have variability in spatial architecture, but it would be considerably reduced compared to the whole variability of parts. Thus, we will privilege 3D Design in context with a context specified by architecture models greatly simplifying the variability selection of the context.

Another important difficulty observed for configurable DMUs is the management the variability of positions of assemblies and parts. For the supply chain, variability management at the BOM level does not need to consider the positions. But DMU has to do it. It is extremely important to use an architecture relative positioning in DMU to minimize the variability. Otherwise, absolute positioning will introduce additional variability even where sub-assemblies are strictly identical, just because they have different absolute positions.

So we recommend modeling spatial architecture of the product with the required variability and based on the following types of models:

- Reference geometry models. They specify dimensions and provide the architecture of positioning through the set of triaxial geometric frame of reference needed to support relative positioning of lower level assemblies.
- Geometric Interface Models. They specify the geometric interface between physical assemblies.
- Space allocation geometry to define the overall space allocation for each of the physical assemblies.

4 System Engineering and variability management

4.1 Impact of systems massive intrusion in traditional products

Systems and software are invading traditional hard products to make them smarter and to allow them to operate as pieces of larger systems. This trend increases the complexity of products. For example, the automotive industry anticipates now an order 10 million of lines of code for the embedded systems of one car. Management of variability needs to cover this systemic dimension. Said in another way, in engineering, the variability of industrial products cannot be managed only under the angle of the physical products structures anymore, as it is often done, but needs to address the system variability (including functional, and behavior). Moreover, these two dimensions of variability need to be managed consistently in configuration (including the management of changes across times).

The PLM/PDM systems role is mainly focused on sustaining the engineering activities for the definition of product families carrying internal variability (design, development and change), as well as the definition of the technical processes of their life-cycle and the definition of the technical resources involved by these processes.

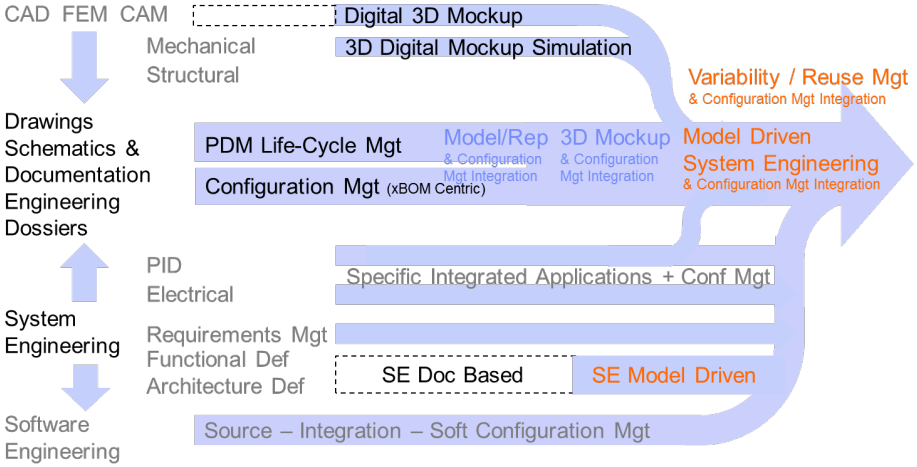
Figure 5 hereunder summarizes the different components of the PLM. It illustrates that the full coverage of all engineering activities relies on different sources of applications which were progressively integrated around PDM systems. It illustrates also that System and Software Engineering are still often being managed independently and reflects a need and a trend to make them converge under a consistent and integrated configuration and variability management

Until now, the PLM/PDM focus was mostly dedicated to managing the technical dossiers (definition, manufacturing and maintenance) and the DMU. This explains that the focus of configuration and variability management in PLM/PDM was to manage the physical product structure and parts configuration.

But there are two strong trends in the PLM/PDM landscape:

- The transformation of system engineering with the development of Model Based System Engineering (MBSE) and a better integration of software engineering.
- The very fast and strong intrusion of systems into traditional products which push PLM/PDM editors to better address and integrate system engineering under the PLM/PDM umbrella.

Figure 5: PLM progressively integrates all engineering specialized applications under the PDM umbrella for configuration, models and documentation management



Sections 4 and 5 discuss the perspectives for using the System Engineering approach for improving reuse and variability management and overcoming current limits observed in current PDM practices to address them.

4.2 Concepts and lessons from Software Variability Management

Variability management has been largely studied in numerous research papers. There are differences between variability management of software and physical products. Software is largely immaterial, easily produced and can be easily changed while physical products often require expensive manufacturing facilities and tools and changes are difficult and costly or even impossible on the already manufactured products. But for many aspects, variability management faces the same challenge in both kind of industries. Moreover, due to the very important intrusion of electronic and software in traditional industry, variability in both domains needs to be managed consistently as the "mechatronics" nature of present products induces dependencies between them. The [Chen, Babar & Ali 2009] paper reviews research studies in variability management and software products lines management (SPL) which is an equivalent concept of products families. [Capilla, Bosch & Kang 2013] made a systematic review of the main concepts and principles used for managing variability. When looking at these papers and others cited in the bibliography, we may notice several interesting concepts, questions and approaches to solve them.

Feature modeling is a key concept to specify and model SPL. SPL is defined from the variability point of view by a feature tree. It structures the configuration model of SPL with two main types of nodes: variability points and, under them, options-variants nodes. Variability points means there is a variability choice to make at this level to configure the software product. The choice must be made by selecting one of the options-variants nodes proposed as child nodes. Option means that the node can be selected or not. Variant means that one of the variant nodes must be selected. [Kang, Cohen, Hess, Novak & Peterson 1990] proposes a method for identifying and specifying features (Feature-Oriented Domain Analysis FODA method). These basic principles are enriched with more possibility of specification of the cardinality of the choices and the possibility to add attributes and constraints in the model [Capilla, Bosch & Kang 2013]

But numerous research papers point out some questions and difficulties and approaches to respond to them. We list hereunder these points and how they are linked to our observations and industry experience in hard products domain.

1. Definition and number of features
   The way to define and choose the features to build the configuration model may be difficult because variability may be seen from different points of view and the number of features to support them may become important and complex to manage. [Capilla, Bosch & Kang 2013] said that features are used by a feature based approach as container of:
   - Capability that is delivered to a customer
   - Requirements containers i.e., units of requirement specifications
   - Product configuration and configuration management
   - Development and delivery to customers
   - Parameterization of reusable assets
   - Product management for different segments

   Difficulties to define variability criteria and values often encountered in the hard products industry would benefit from an approach based on requirements and feature modeling.

2. Multiple point of views for variability:
   One of the reason for complexity of features modeling is the fact that the variability modeling must endorse different points of view. [Chen, Babar & Ali 2009] underlines first a distinction between external variability (as seen externally by the customer) and the internal variability or technical variability. It also shows that requirements are progressively defined and refined from the initial architecture definition stage to the running system across all the stages of software development.
   This is a current weakness of variability management in the hard product industry to essentially manage the physical product point of view and not the others.

3. Variability and product / systems life cycle definition artifacts:
   The definition of a complex systems or products is made through artifacts organized in different hierarchical structures, often managed relatively separately. The variability model has to be declined on each of these structures to retrieve and compose them accordingly to the configuration selected. [Jiao & Tseng 1999] addresses this topic by proposing an integrated data model mixing the different views consistently (not specific to software engineering). [Asikainen, Soininen & Männistö 2003] studies and compares how applications used to model and manage software architecture may be also used to manage variability through product configuration. This is also comparable to the industry experience presented in section 3 where we see how PLM is used to manage the variability, but with a focus on physical product structure and DMU.

4. Multilevel variability – Multi-level product families
   [Reiser 2009], who deeply analyzes SPL for automobile, suggests that variability must be designed at different levels. This fits with our observation and approaches developed in section 3.2 well (multi-level products families). This requirement seems to be fundamental if we want to reuse a variable module in different variable platforms and in different variable vehicles. [Reiser 2009] develops a concept of configuration link which seems close to the concept of variability-effectivity mapping that we describe in section 3.2 and whose mechanism is provided by some PLM/PDM software packages. This approach and principle should also enable "local" specification and to management of the variability by considering only those which are meaningful for the perimeter of the considered product family. Inside the low level product family reused, the variability effectivity definition is not constrained to be expressed by options/variants values of the higher level product families. Configuration links or equivalent variability-effectivity mapping rules are needed to select the proper lower level family configuration corresponding to the variability effectivity of the upper level. Another way to see it is that this multi-level approach may enable some decoupling of the specification of the variability of the upper level product family from that of the lower level reused product family. This make possible to hide internal complexity of the variability lower level product family from the upper one. The classic example for automotive is to decouple a commercial feature such as "level of equipment" (values: lux – comfort – economy) and "country" from the technical features used at a module level such as a seat. A technical feature at the seat level such as "heating seat", for example, could be linked through a configuration link/mapping rules to the equipment level "lux" in Southern Europe and "comfort" in Northern Europe.

5 Conclusions: Using the System Engineering approach to define and model Systems/Products Families variability

These are just embryonic and not yet proven ideas which came to me when confronting my experience and the research review I have made in SPL. These ideas are driven by the conviction that the system engineering and the traditional physical product engineering approaches need to be consistently integrated into a unified approach and model.

From the experiences seen and described here, we may derive some conclusions and intuitions for the future:

1. It is necessary to find a common approach for managing systems variability and product variability. Until now, the two approaches were traditionally managed separately. The hard products industries mainly focused their PDM and ERP systems on managing the physical product variability. Strong intrusion of systems in hard products industry and growing interdependency of functional and physical dimensions push for an integrated approach
2. Variability criteria (features) are strongly related to requirements. In other words, it seems us that a variability criterion value may be quite formally linked to a consistent set of requirements.
3. The system model (according to SYSML common standard) offers a way to simultaneously and consistently mix the requirements, the functional, the physical and the behavioral points of view. So, if we are able to model the variability through a consistent set of features (requirements regrouping in line with systems model components), we have a solution to the question of multiple points of view for variability.
4. System level requirements are defined at the beginning of the system design and refined and allocated to the system architecture components in parallel with the architecture design. So variability definition is naturally and strongly related to the system engineering approach. Adding a feature concept to SYSML model (functional, physical and behavioral) offers a perspective to model variability progressively with the system architecture development, and to manage consistently the different points of view provided by SYSML.
5. Variability must be multi-levelled and structured by an architectural approach. When defining a system level, it is only required to define or know the external specifications of the sub-systems it relies on, but it is not required to define them internally. This abstraction capability enables us to define the variability focusing the engineering effort for the relevant level of abstraction. The system engineering approach could strongly help to properly define the architecture of a complex system/product in a hierarchy of product families according reuse and variability strategy. Moreover, limiting the variability of the architecture itself by standardizing interfaces may enable us to fit in different module families without (or limited) side effects on neighbor modules it interfaces with.

There are still questions not addressed here, which would need to be studied, for example:

- How, with a systemic approach, would we model manufacturing resources and processes (with the linkage of process to the product and with the variability at all levels product, process and resources)?
- What does variability for behavior mean (simulation, test, validation)? What variability in behavior is driven by the system or product variability definition, and what variability is added by the methods and process for simulating, testing and validating?

Nevertheless, for the reasons exposed above, it looks to us that the system engineering approach and that system-based-modeling-engineering (SBME) relying on SYSML could be a strong foundation for supporting the definition of complex systems/products with their variability. The configuration-linking/variability-effectivity-mapping-rules needs to be articulated with the system/sub-system concept. In this perspective, classic product definition would be embedded into the system definition. This model would also need integration of proper positions management with the physical description of the system and with the system/sub-system architecture.

I would be interested in getting feed-back of researchers about these ideas and possible research done on this subject I may have missed.

References

[Asikainen, Soininen & Männistö 2003] Towards Managing Variability using Software Product Family Architecture Models and Product Configurators
Timo Asikainen, Timo Soininen, Tomi Männistö – Helsinki University of Technology – Software Business and Engineering Institute (2003 SoberIT).
http://www.soberit.hut.fi/pdmg/papers/ASIK03TOW.pdf

[Becker 2003] Towards a General Model of Variability in Product Families
Martin Becker -System Software Group, University of Kaiserslautern
Kaiserslautern, Germany - mbecker@informatik.uni-kl.de

[Capilla, Bosch & Kang 2013] Systems and Software Variability Management - Concepts, Tools and Experiences -
Capilla, Rafael, Bosch, Jan, Kang, Kyo-Chul (Eds.). SPRINGER
http://www.springer.com/computer/swe/book/978-3-642-36582-9

[Chen, Babar & Ali 2009] Variability management in software product lines: a systematic review,
Proceeding- Lianping Chen (University of Limerick, Ireland) - Muhammad Ali Babar (University of Limerick, Ireland) - Nour Ali (University of Limerick, Ireland) - SPLC '09 Proceedings of the 13th International Software Product Line Conference - Pages 81-90 -Carnegie Mellon University Pittsburgh, PA, USA ©2009
http://dl.acm.org/citation.cfm?id=1753247

[Jiao, Simpson, Siddique 2007] Product family design and platform-based product development: a state-of-the-art review
Jianxin (Roger) Jiao - Timothy W. Simpson Zahed Siddique - Springer Science+Business Media, LLC 2007 - July 2007

[Jiao & Tseng 1999] An Information Modeling Framework for Product Families to Support Mass Customization Manufacturing
Jianxin Jiao, Mitchell M. Tseng)
Department of Industrial Engineering and Engineering Management, The Hong Kong University of Science and Technology, Kowloon, Hong Kong
CIRP Annals-Manufacturing Technology, 1999 – Elsevier

[Kang, Cohen, Hess, Novak, Peterson 1990] Feature-Oriented Domain Analysis (FODA) Feasibility Study
Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak,
A. Spencer Peterson
Software Engineering Institute - Carnegie Mellon University - Pittsburgh, Pennsylvania 15213
November 1990
http://www.sei.cmu.edu/reports/90tr021.pdf

[Lim 1994] Effect of reuse on quality, productivity and economics
WC. Lim Hewlett Packard– IEEE Software September 1994

[Manet Hamm O.Brien 2011] Making the word work better – The Ideas that shaped a century and a company 'IBM)
Kevin Manet, Steve Hamm, Jeffrey M.O'Brien
IBM Press/ Pearson 2011

[Pil, Holweg 2004] Linking Product Variety to Order-Fulfillment Strategies
Frits K. Pil – University of Pittsburgh - Matthias Holweg - Judge Institute of Management, University of Cambridge –
Interfaces 2004 – Vol 34, N° 5 September-October pp 394-403

[Reiser 2009] Managing Complex Variability in Automotive Software Product Lines with Subscoping and Configuration Links
Mark-Oliver Reiser Fakultäat IV {Elektrotechnik und Informatik der Technischen Universitäat Berlin

[Renault Nissan 2013] Common Module Family (CMF): a new approach to engineering for the Renault-Nissan Alliance
http://media.renault.com/global/engb/alliance/Media/PressRelease.aspx?mediaid=49441

[Rokunuzzaman & Choudhury 2006] Economics of Software Reuse and Market Positioning for Customized Software Solutions
M. Rokunuzzaman+ and Kiriti Prasad Choudhury+*
*School of Engineering & Computer Science, Independent University, Bangladesh (IUB), Dhaka, Bangladesh and Beximco Pharmaceuticals Ltd, Dhaka, Bangladesh. zaman.rokon@yahoo.com, kpmoni@yahoo.com
Journal of Software, Vol 6, n°1, January 2006

[Simpson, Siddique & Jiao 2007]
Platform-Based Product Family Development
Timothy W. Simpson[1] -, Zahed Siddique[2,] and Jianxin (Roger) Jiao[3]
[1]Departments of Mechanical & Nuclear Engineering and Industrial & Manufacturing - Engineering, The Pennsylvania State University, University Park, PA 16802;
[2]School of Aerospace and Mechanical Engineering, University of Oklahoma, Norman, OK 73019;
[3]School of Mechanical and Aerospace Engineering, Nanyang Technological University,Singapore 639798 - Springer 2007

[Volkswagen 2011] Volkswagen Fact book 2011
http://www.volkswagenag.com/content/vwcorp/info_center/de/publications/2011/04/Volkswagen_Group - Factbook_2011.bin.html/binarystorageitem/file/Factbook+2011.pdf

[Zhang & Fan 2006] A Conceptual Framework for Product Lifecycle Modeling

Wenlei Zhang[1,2], Yushun Fan[3]
[1]Shenyang Institute of Automation, Chinese Academy of Science, Shenyang, P.R. China
[2]Gradute School of the Chinese Academy of Science, Beijing, P.R. China
zwl@sia.cn
[3]Dept. of Automation, Tsinghua University, Beijing, P.R. China
fanyus@tsinghua.edu.cn
Innovative Computing, Information and Control 2006 ICICIC '06 Vol 2

[Zhang & Fan 2007] A Conceptual Framework for Product Lifecycle Modeling

Wenlei Zhang[1,2], Yushun Fan[3]
[1] Shenyang Institute of Automation, Chinese Academy of Science, Shenyang, P.R. China
[2] Gradute School of the Chinese Academy of Science, Beijing, P.R. China - zwl@sia.cn
[3] Dept. of Automation, Tsinghua University, Beijing, P.R. China - fanyus@tsinghua.edu.cn
J Intell Manuf (2007) 18:5–29 - DOI 10.1007/s10845-007-0003-2

Accounting for Uncertainty and Complexity in the Realization of Engineered Systems

Warren F. Smith[1], Jelena Milisavljevic[2], Maryam Sabeghi[2], Janet K. Allen[2], and Farrokh Mistree[2]

[1] School of Engineering and IT, University of NSW Canberra, ACT, Australia
[2] Systems Realization Laboratory**,** University of Oklahoma**,** Norman, OK, USA

Abstract. Industry is faced with complexity and uncertainty and we in academia are motivated to respond to these challenges. Hence this paper is the product of thoughts for exploring the model-based realization of engineered systems. From the perspective that the activity of designing is a decision making process, it follows that better decisions will be made when a decision maker is better informed about the available choices and the ramification of these choices. Presented in this paper, in the context of an example of designing a small thermal plant, is a description of an approach to exploring the solution space in the process of designing complex systems and uncovering emergent properties. The question addressed is that given a relevant model, what new knowledge, understanding of emergent properties and insights can be gained by exercising the model?

Keywords: Decision-based, Model-based, Compromise, Complex Systems, Solution Space Exploration, Decision Support Problem

1. MOTIVATION

Designing, in an engineering context, is an activity that seeks to deliver a description of a product to satisfy a need in response to a stated objective and/or set of requirements. In the process, it may involve invention and/or the application of science and engineering knowledge to resolve a solution. Given that multiple solutions may be proposed with differing measures of merit, it follows that the paramount role of a designer is that of a decision maker. It is further argued that understanding the inherent choices and risks within the context of a design lead to justifiable decisions. In an age where issues such as efficiency, equity, sustainability and profitability are equally valid decision drivers the motivation to develop theories and approaches to explore the design and aspiration spaces is strong. Indeed, this is what motivates the academic design community in general and the authors of this paper in particular.

2. FRAME OF REFERENCE

Design choices can be explored through first building sufficiently detailed and valid mathematical models, and then exercising these models and seeking understanding of their behavior and the emergent properties (those that are unforeseen and influenced by uncertainty). Such models can very quickly become very complicated. Organized complexity in the context of systems theory is said to arise from the combination of parts that form a system but the behavior of the system is not necessarily controllable or predictable from knowledge of the parts alone. Disorganized complexity in contrast is a reflection of the random and statistical variability of the parts and the subsystems and system they form. It follows that to grow complex system knowledge requires the management of aspects of both complication (complexity) and uncertainty. Managing uncertainty raises concerns such as those due to the imprecise control of process parameters, the incomplete knowledge of phenomena, the incomplete models and information aggregation, and the need to explore alternatives. Managing issues of complication include dealing with the trade-off between accuracy and computational time, the levels of interdependencies between parts, and the allocation of resources to exploring the solution and aspiration spaces. It follows that the challenge for engineers is the creation of knowledge about the system and the challenge encompasses capturing tacit knowledge, building the ability to learn from data and cases, and developing methods for guided assistance in decision making.

The authors have adopted a model based approach in pursuing these challenges recognizing that models can have different levels of fidelity, they can be incomplete and possibly inaccurate (particularly during the early stages of design).

2.1 The Decision Support Problem

Used is the Decision Support Problem (DSP) construct that is based on the philosophy that design is fundamentally a decision making and model-based process [1, 2]. A tailored computational environment known as DSIDES has been created as an implementation of the method. The DSP and DSIDES are well documented in [3-7].

Reported applications of this approach include the design of ships, damage tolerant structural and mechanical systems, design of aircraft, mechanisms, thermal energy systems, composite materials and the concurrent design of multi-scale, multifunctional materials and products. A detailed set of early references to these applications is presented in [8]. Key applications more recently span specification development [9, 10], robust design [11-14], product families [15-17], the integrated realization of materials and products [18-22], and a variety of mechanical systems [23-26].

The nature of a decision and model-based approach to designing through modelling the physical world is portrayed in Figure 1. Once a model is appropriately formulated, DSIDES, with its operations research tools (traditionally an adaptive sequential linear programming algorithm delivering vertex solutions), is used to deduce "model conclusions" [5]. Where dilemmas exist this process may be iterative in nature and demand significant justification. It thus becomes imperative to be able to describe and understand the design and aspiration spaces and to be able to explore these spaces.

Key is the concept of two types of decisions (namely, selection and compromise) and that any complex design can be represented through mathematically modelling a network of compromise and selection decisions [4, 6]. Being able to work with the complexity of these decision networks is also a foundational construct as are the axioms of the approach as detailed in References [4, 6].

In reflecting on the compromise DSP, parallels with the "demands" and "wishes" of Pahl and Bietz [27] can be drawn. The demands are met by satisfaction of the DSP constraints and bounds and the wishes are represented by the goals. Collectively, the constraints and bounds define the feasible design space and the goals define the aspiration space. The feasible and aspiration spaces together then form the solution space. Note that a selection DSP can be formulated as a compromise DSP [28] where the key words "Given", "Find", "Satisfy" and "Minimize" are used.

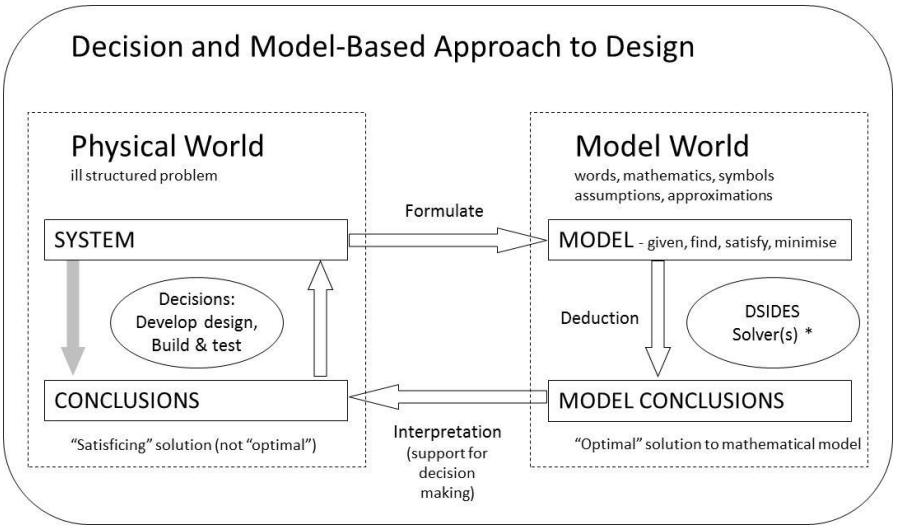


Fig. 1. Modelling the Physical World

2.2 Understanding the Solution Space

A strategy for identifying a possible solution space and exploring it using tools within DSIDES includes:

- Firstly, discover regions where feasible designs exist based on satisfying the constraints and bounds or where they might exist by minimizing constraint violation.
- Secondly, from the neighborhood of feasible or near feasible regions frame the feasible design space extremities using a preemptive (lexicographic minimum) representation of the goals in a higher order search.
- Thirdly, having framed the space and the zones of greatest interest, move between the extremes generating deeper understanding and exploring tradeoffs using an Archimedean (weighted sum) formulation of the goals.

Our focus in this paper is on the first two steps. To discover feasible regions, zero, first and second order methods are currently available in DSIDES.

This overall process is conceptually reflected in Figure 2 where over time knowledge, confidence and utility increase while converging to a recommended decision. The decisions are made through a series of diverging, synthesizing and convergent decision making processes. As will become clearer, various tools may be used to support different decisions.

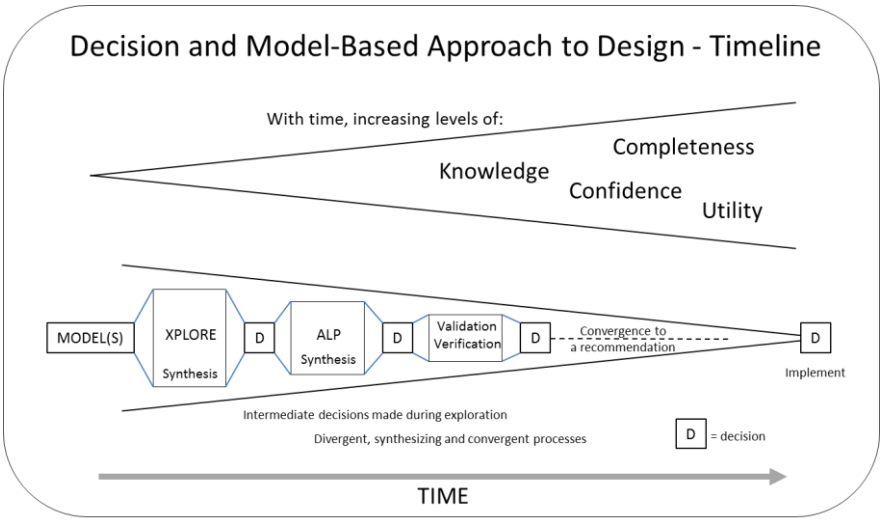


Fig. 2. Modelling and Decision Timeline

The most rudimentary approach within DSIDES is a zero order search referred to as XPLORE. Based on the algorithm of reference [29], it is used to test a range of designs within the stated system variable bounds. The best N designs are kept providing candidate starting points for higher order searches. A second method utilizing a pattern search algorithm is also available within the INITFS (Initial Feasible Solution) module. Used in series, these methods can assist greatly in delivering the Adaptive Linear Programming (ALP) algorithm a starting point from which the likelihood of achieving greater understanding of the solution space is high. In the case of a multimodal solution space a variety of starting points are employed.

Various methods may be applied to conduct post solution analysis on the data generated including visualization through the use of various plots. Given that in ALP is a linear based simplex solver, the opportunity to explore sensitivity using primal and dual information exists. Also provided in DSIDES is information about the monotonic characteristics of the model. In concert, all these elements contribute to the effective modelling, framing, exploration and dilemma resolution that is necessary when considering the design of complex systems.

3. SCENARIO FOR THE EXAMPLE

The study at the core of this paper is being developed to support the growing research effort within the Systems Realization Laboratory at the University of Oklahoma. Current research interests in the laboratory inter alia span complex systems, dilemma management, design space modelling and exploration, post solution analysis, and sustainability when considering economic, socio-cultural, and environmental issues. One domain allowing all these matters to be explored is thermal systems.

There are many possible applications for small scale "power" plant systems that make direct mechanical use of the power produced or that run small generators to produce electricity. Examples include provision of power to equipment in farming irrigation systems, driving reverse osmosis systems to produce fresh water for remote communities and generating electricity for general use in small collectives in both 1st and 3rd world environments.

A common approach given an available heat source is to build such a system around the Rankine cycle, a mathematical representation of a "steam" operated heat engine. A schematic representation of the Rankine cycle is shown in Figure 4 where the primary components of the system are a power producing turbine, a pump to pressurize the flow to the turbine and two heat exchangers; a condenser and a heater. In the context of building a model using a decision-based approach to design, such a thermal system affords complexity to be developed and dilemmas to be managed and resolved, both hypothetically and practically. Modelling the Rankine cycle represents Stage 1 of the model development and will be referred to herein as the foundational example model. Future expansion within the laboratory will deal with heat source issues (to the left in Figure 3) and power use issues (to the right in Figure 3) and the choice of working fluids. The common working fluid in a Rankine cycle is water. Uses of other fluids (often organic in chemistry) have given rise to the development of "organic Rankine cycles". Of course geometric specification and design analysis of physical elements in the system also represent opportunities for model and design space exploration.

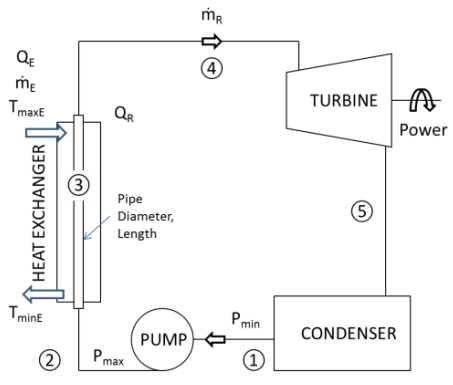


Fig. 3. Stage 1 Model Schematic

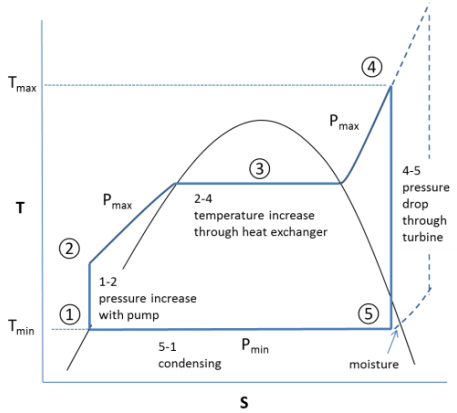


Fig. 4. Rankine Cycle (Temperature v Entropy)

4. THE FOUNDATIONAL EXAMPLE MODEL

The foundational example model is defined by the cycle's maximum and minimum pressures and maximum temperature (PMAX, PMIN and TMAX). Energy is transferred to the closed loop Rankine cycle through a heat exchanger. The heat exchanger is assumed to be of a counter flow design where the key characteristic is the maximum temperature of the heating flow (TMAXE).

From a decision based design approach, the determination of satisficing[1] values of these variables represents a coupled compromise-compromise DSP dealing with the Rankine cycle (PMAX, PMIN and TMAX) and the heat exchanger (TMAXE) respectively. Two additional decisions have been built into the template of the current model, namely, the selection of the fluids for both the heating and Rankine cycle loops. Therefore, in concept the current model is a compromise-compromise-selection-selection problem. Further complexity in the model will be developed in due course to reflect aspects of the mechanical design of the system components (eg., dimensions).

The ideal Rankine cycle involves 4 processes, as shown graphically in the Temperature (T) versus Entropy (S) plot in Figure 4. There are two adiabatic isentropic processes (constant entropy) and two isobaric processes (constant pressure).

Referring to Figure 4,

①-② adiabatic pumping of the saturated liquid from PMIN to PMAX
②-④ isobaric heat addition in heat exchanger to TMAX,
④-⑤ adiabatic expansion in the turbine from PMAX to PMIN producing power with the possibility of wet steam exiting the turbine, and
⑤-① isobaric heat loss in the condenser.

The isothermal segments represent moving from saturated liquid to saturated vapor in the case of ③ in the heater and the reverse in the condenser between ⑤-①. The key thermodynamic properties of the working fluid(s) are determined using REFPROP [30]. For the purposes of this paper focus has been placed on the compromise-compromise aspects and a number of system variables have been treated as parameters. One such simplification is the use of water as the working fluid in both loops.

The combined model may be summarized using the compromise key words as:

GIVEN

Water as the fluid in the Rankine cycle
Water as the heat transfer medium in the heat exchanger
The minimum pressure in the Rankine cycle (PMIN – defined as a parameter)
Ideal Rankine cycle thermodynamics
Ideal heat transfer in the heat exchanger
Thermodynamic fluid properties (determined using REFPROP)

[1] *Satisficing* is a decision-making strategy or cognitive heuristic that entails searching through the available alternatives until an acceptability threshold is met. This is contrasted with optimal decision making, an approach that specifically attempts to find the best alternative available. Wikipedia.

FIND
x, the system variables
PMAX Maximum pressure in the Rankine cycle
TMAX Maximum temperature in the Rankine cycle
TMAXE Maximum temperature of the heating fluid
d^- and d^+, the deviation variables
SATISFY
The system constraints:
Temperature delta for maximums in exchanger
Moisture in turbine less than upper limit
Rankine cycle mas flow rate less than upper limit
Temperature at ④ ≥ temperature at ③
Quality at ④ is superheated vapor
TMAXE greater than TMINE by at least TDELE
TMINE ≥ temperature at ② by at least TDELC
Ideal Carnot cycle efficiency greater than system efficiencies (sanity check)
Temperatures within valid ranges for REFPROP fluid database
The system variable bounds ($x_j^{min} \leq x_j \leq x_j^{max}$):
$500 \leq PMAX \leq 5000$ (kPa)
$350 \leq TMAX \leq 850$ (K)
$350 \leq TMAXE \leq 850$ (K)
The system goals:
Achieve zero moisture in steam leaving the turbine (ie., steam quality of 1)
Maximize Rankine cycle efficiency where RCEFF = $(P_{turbine} - P_{pump})/Q_{in}$
Maximize temperature exchanger efficiency
where TEFFEX = (TMAXE-TMINE)/(TMAXE-TEMP2)
Maximize system efficiency indicator 1 where SYSEF1 = $(P_{turbine} - P_{pump})/Q_{out}$
Maximize system efficiency indicator 2 where SYSEF2 = RCEFF*TEFFEX
Maximize heat transfer effectiveness in exchanger
where HTEFF = f(heat transfer coefficient, geometry, flow rates etc.)
MINIMIZE
The deviation function, $Z(\mathbf{d^-}, \mathbf{d^+}) = [f_1(\mathbf{d^-}, \mathbf{d^+}), \ldots, f_k(\mathbf{d^-}, \mathbf{d^+})]$
where the deviation function is expressed in a preemptive form.

The six system goals in the example have been placed at six levels of priority in the implemented preemptive model. The implication is that the first level goal function will be satisfied as far as possible and then while holding it within a tolerance; the second level goal function will be addressed. When the second has been so conditionally minimized it will be held within its tolerance and then the third goal will be worked upon; and so on in an attempt to address all the goals across all levels. Achieving satisfaction of the higher priority goals may cause the sacrifice of achievement of the lower priority goals. By prioritizing the goals differently, comparison may show competing goals driving the solution process in different directions. By grouping more than one goal at the same level, an Archimedean (weighted sum) approach can be accommodated.

5. VALIDATION OF THE MODEL

Structural validity as it applies to a computer code infers that the logic and data flows between modules are correct. This does not guarantee accuracy. Performance validity is associated with the accuracy of the results achieved as measured against reliable benchmarks and/or reasoned argument (other published work, known physical characteristics etc.).

5.1 Structural Validity of the Model

The compromise DSP is a hybrid multi-objective construct and this approach to designing has been validated through use [6]. The primary solver in DSIDES is an Adaptive Linear Programming algorithm, and it has also been described and validated elsewhere [5]. The current instantiation has also been shown to replicate some standard test problems. The REFPROP database is a key thermodynamic property model from NIST [30] and the NIST supplied subroutines and fluid files have been used. The total system has been integrated in a FORTRAN environment using G FORTRAN compilers on a PC platform. The functioning of the code has been successfully demonstrated to reproduce results consistent with text books and other programs providing thermodynamic properties of fluids.

Consistency and logical relationship between the constructs were checked by testing several inputs and reviewing the expected outputs, e.g., thermodynamic properties of water at different pressures and temperatures.

5.2 Performance Validity of the Model

Performance validity was checked through exercising the thermal model, i.e., investigation of the model by parametric study such as net power output. For instance, since the power is a function of Rankine flow rate, it is expected that higher flow rates are necessary to produce higher power. This was verified and is discussed in Section 6.

The next step for performance validity of the model was through checking the behavior of the goals. This model includes six goals, five of which estimate measures of efficiency: the Rankine cycle efficiency, the heat exchanger efficiency, two formulations of system efficiency and the heat exchanger effectiveness. By exploring different possibilities in the goal priorities for the example and by examination of the monotonicity of the goals [31] it was discovered that the prioritization of the efficiency goals in a preemptive formulation will drive the system in two directions.

If prioritization is given to the Rankine cycle efficiency and/or system efficiency formulation 1 the solutions are of high temperature and high pressure character. In discussing the results this ordering of priority will be referred to as “Order 1”. In contrast, low temperature and low pressure solutions are preferred if the heat exchanger efficiency, system efficiency formulation 2 and/or heat transfer effectiveness are prioritized (Order 2). This behavior of the model is appropriate and predictable given the model goal formulations.

6. DISCUSSION OF RESULTS

Consider that a plant producing a baseline of 25kW is required and that higher powers are sought but the maximum steam that can be produced is 0.1 kgs^{-1}. What are the characteristic values that define the Rankine cycle and the heat exchanger?

In answering this question, a two-step process using DSIDES is used, firstly with the XPLORE grid search module and then with the ALP algorithm.

As described in Section 4, variable bounds have been defined but do they encompass feasible designs? Using XPLORE, this question is examined. Presented in Figure 5 is a plot of TMAX versus PMAX showing discrete tested combinations that lead to feasible designs for 25, 50 and 70kW cases. Feasible designs exist where the constraint violation is zero. The extent of the plot reflects the bounds of each system variable. The contraction in the number of designs and the size of the design space at least in the two dimensions shown as power increases is clearly evident. The area covered by these can be interpreted as being representative of the feasible design space(s).

Further use of EXPLORE can and has in this example been made to examine the regions where goals are fully satisfied or at least minimized. Being keen to ensure longevity of the plant, the operational requirement is that moisture in the steam exiting the turbine is minimized. Therefore, the Level 1 priority goal for all results presented is that of minimizing moisture. If this were the only goal specified it can be shown as in Figure 6 that there are many designs that could achieve less than 5% moisture while producing 25 kW or 50 kW. Shown in Figure 7 are those designs with zero percent moisture.

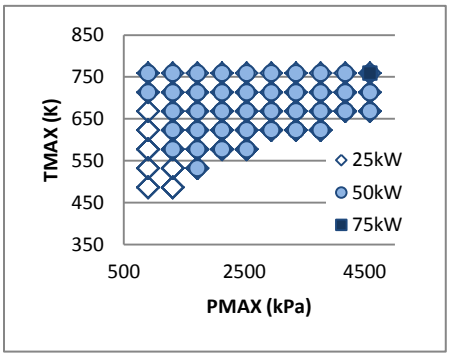


Fig. 5. Feasible designs using XPLORE (less than 12% moisture)

It follows that other goals need to be subsequently specified to achieve singular (local) convergence. For the 25 kW designs, using the XPLORE data, if some moisture is allowed (up to 12%) higher Rankine cycle efficiencies can be achieved with designs depicted in the region shown in top right of Figure 8 (efficiencies better than 27.5%). However, constraining the designs to have zero moisture caps the best Rankine cycle efficiency found at 25% (PMAX 2136 kPA and TMAX 759 K), signifi-

cantly to the left of the Figure 8 cluster. This reflects the best "Order 1" XPLORE solution.

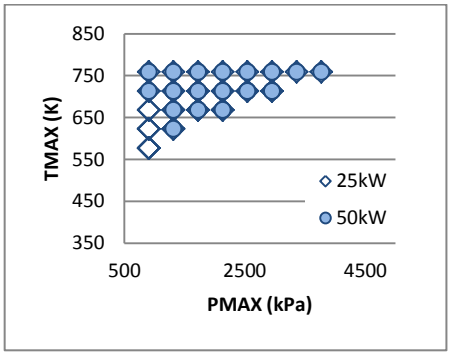


Fig. 6. Feasible designs with moisture less than 5% using XPLORE

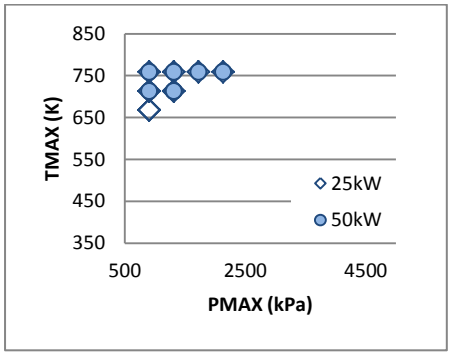


Fig. 7. Feasible designs with 0.000% moisture using XPLORE

Considering the second system efficiency goal representation, SYSEF2, if set as priority one, values of 16% in the lower left region shown in Figure 8 are possible. If, constraining the designs to have zero moisture caps the best SYSEF2 value found is 12% (PMAX 909 kPA and TMAX 668 K), significantly higher than the Figure 8 cluster. This reflects the best "Order 2" XPLORE solution.

To summarize, higher Rankine cycle efficiencies are achieved with high temperatures and high pressures. In contrast, the higher system efficiencies result from low temperatures and low pressures. And, to achieve zero moisture in the turbine, the requirement is for high temperatures with lower pressures. Clearly, the right decision is not straightforward.

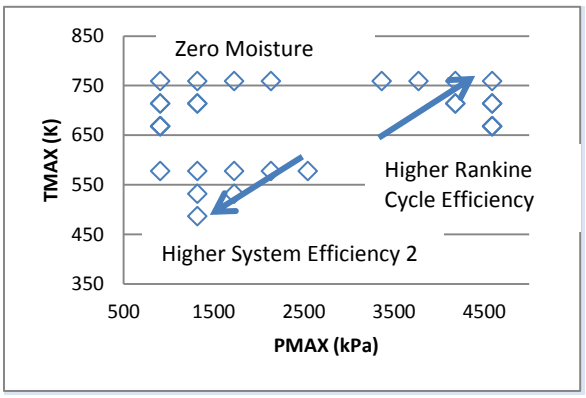


Fig. 8. Trade-offs for feasible designs for 25kW using XPLORE (less than 12% moisture)

While the framing value of using the XPLORE DSIDES module has been demonstrated, what further insights can be developed using the DSIDES ALP algorithm [5] to refine understanding?

The next set of results presented are for the two groupings of the goals as discussed in Section 5, one producing high temperature and pressure results (Order 1) and the other low temperatures and pressures (Order 2). The goal deviation variable values associated with each goal (defined in Section 4) have been named with a leading "G", for example GRCEFF referring to the Rankine Cycle Efficiency goal.

Given an upper limit on the mass flow rate in the Rankine cycle of 0.1 kgs^{-1}, a parametric study has been undertaken to establish the power output limit for the system. Shown by the results tabulated in Table 1 (for both Order 1 and Order 2), are solutions for 25, 50 and 75 kW configurations. While not shown in Table 1 to maintain clarity, for each of the six arrangements (combinations of power output and goal priority order) different starting points were tried yet the solutions for each power output were for all intents and purposes the same, suggesting, though not guaranteeing, that the global minima (for the formulation) may have been found.

Table 1. Parametric Study of Power

		ORDER 1 Priority 1,2,4,3,5,6				ORDER 2 Priority 1,5,3,6,2,4			
		25 kW	**50 kW**	**75 kW**	NOTES	**25 kW**	**50 kW**	**75 kW**	NOTES
INITIAL SOLUTION	PMAX (kPa)	2000	2000	4250		1250	1250	4250	
	PMIN (kPa)	100	100	100		100	100	100	
	TMAX (K)	767	767	767		683	683	767	
	TMAXE (K)	808	808	808		725	808	808	
	ELEN (m)	154	154	154		113	154	154	
	EDIA (mm)	20	20	20		20	20	20	
FINAL SOLUTION	**PMAX (kPa)**	3415	3415	3417	*Consistently high*	826	1287	2889	*Lower, increases with Power*
	TMAX (K)	840	840	840	*Consistently high*	613	743	810	*Lower, increases with Power*
	TMAXE (K)	850	850	850	*Consistently high*	623	753	820	*Lower, increases with Power*
	ELEN (m)	50	50	50	*Insensitive to ELEN*	50	50	50	*Insensitive to ELEN*
GOALS (Deviations)	G1 RCMIT	0.000	0.000	0.000	*Zero Moisture in all*	0.000	0.000	0.000	*Zero Moisture in all*
	G2 RCEFF	0.709	0.709	0.709		0.830	0.780	0.723	
	G3 TEFFEX	0.720	0.430	0.120		0.020	0.010	0.010	
	G4 SYSEF1	0.709	0.709	0.709		0.830	0.780	0.723	
	G5 SYSEF2	0.918	0.833	0.745		0.833	0.783	0.727	
	G6 HTEFF	0.007	0.001	0.000		0.000	0.000	0.000	
Derived Values	**FLOWR (kgps)**	0.027	0.050	0.080	*Comparitively "low"*	0.050	0.080	0.090	*Comparitively "high"*
	RCEFF	0.291	0.291	0.291	*Comparitively "high"*	0.170	0.220	0.280	*Comparitively "low"*
	TEFFEX	0.281	0.573	0.876	*Comparitively "low"*	0.980	0.990	0.990	*Consistently high*
	SYSEF1	0.291	0.291	0.291	*Comparitively "high"*	0.170	0.219	0.277	*Comparitively "low"*
	SYSEF2	0.081	0.167	0.255	*Comparitively "low"*	0.167	0.216	0.273	*Comparitively "high"*
	HTEFF	0.990	0.990	1.000	*Stable - Ideal assumption*	1.000	1.000	1.000	*Stable - Ideal assumption*
	CARNOT	0.556	0.556	0.556	*Theoretical MAX*	0.392	0.498	0.540	*Theoretical MAX*

The behavior of the model can be assessed in a number of ways including convergence of the system and deviation variable. For the benchmark 25 kW cases, the convergence history for Order 1 is presented in Figures 9 and 10 and for Order 2 in Fig-

ures 11 and 12. All curves reach a stable final steady state. In the case of Order 1, zero moisture in the turbine was not achieved until iteration 9. This aspect dominated the solution process to this point. However, GRCEFF and GSYSE1 which are superimposed are seen to be generally decreasing. The reverse is true for Order 2. For Order 2, zero moisture was achieved from iteration 5 from which point reductions in GSYSE2, GEXEFF and GHTEFF are evident. Clearly an indicator of excess capacity in considering the baseline 25 kW case is that the flow rate in the turbine is well below the defined bound on this variable of 0.1. In framing and exploring a design model, the nature of the specified variable bounds needs to be understood. Some are set based on true physical constraints and some are arbitrary.

The parametric study of power has provided the flow rate results depicted in Figure 13. For Order 1 where Rankine cycle efficiency is favored, the flow rate is lower because of the improved efficiency. Extrapolating to where both flow rate curves would intersect the 0.1 kgs^{-1} upper bound, it would appear that approximately 90 kW would be available in the modelled ideal system. A companion plot of the Rankine cycle efficiency versus power is given in Figure 14 where a consistently high efficiency is achieved for Order 1. The efficiencies produced under Order 2 are forced to increase in order to produce the higher power demands. In contrast, the final plot presented, Figure 15, is used to highlight that by prioritizing the goals as per Order 2, higher values of system efficiency as measured by the second formulation can be achieved. This formulation is a product of the efficiencies of the two primary system components, exchanger and Rankine cycle. Because of the idealized efficiency of the exchanger being higher than that of the Rankine cycle, this term dominates and therefore drives the solution to the lower temperatures and pressures that suit the exchanger. The monotonically increasing curves of Figure 15 further suggest that higher overall efficiencies will come with higher power.

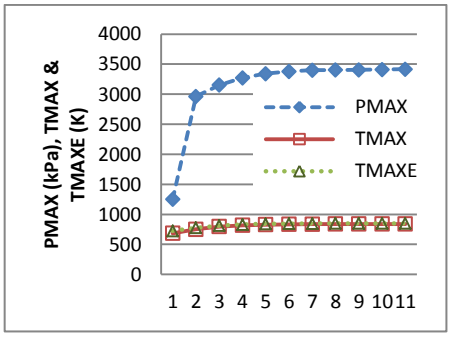


Fig. 9. Order 1 system variable (25kW) convergence plotted against iteration,

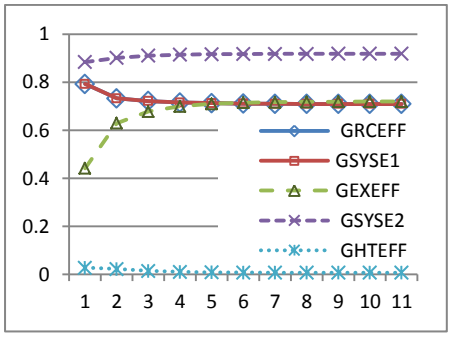


Fig. 10. Order 1 deviation variable (25kW) convergence plotted against iteration, (lower values preferred, GRCEFF and GSYSE1 superimposed)

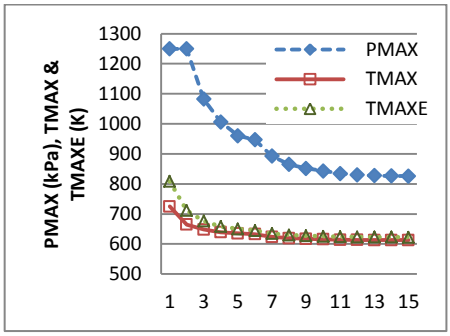


Fig. 11. Order 2 system variable (25kW) convergence plotted against iteration,

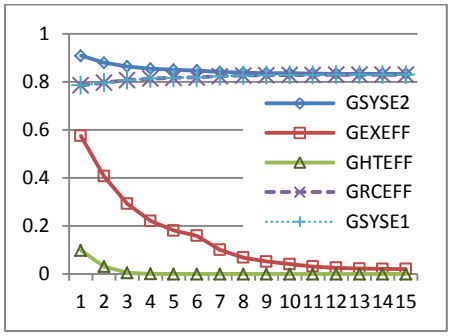


Fig. 12. Order 2 deviation variable (25kW) convergence plotted against iteration, (lower values preferred, GRCEFF and GSYSE1 superimposed)

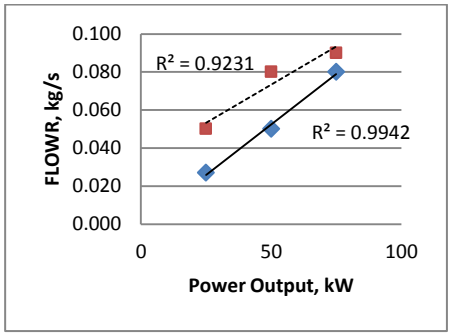


Fig. 13. Rankine Cycle Mass Flow Rate, FLOWR versus Power Output (Order 1 – solid line; Order 2 – dashed line)

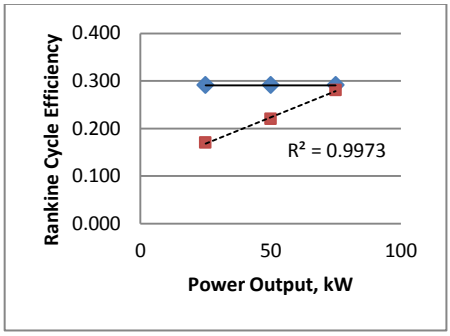


Fig. 14. Rankine Cycle Efficiency versus Power Output (Order 1 – solid line; Order 2 – dashed line)

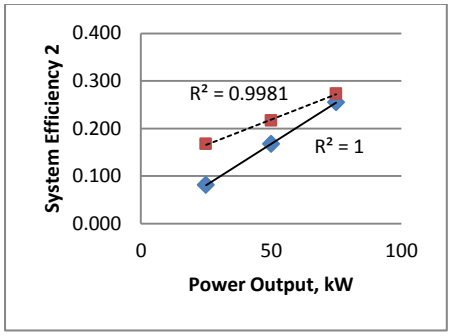


Fig. 15. System Efficiency 2, GSYSE2, versus Power Output (Order 1 – solid line; Order 2 – dashed line)

7. CONCLUDING REMARKS

Industry is faced with complexity and uncertainty and we in academia are motivated to respond to these challenges. Hence this paper is the product of thoughts for exploring the model-based realization of engineered systems. What new knowledge, understanding of emergent properties and insights can be gained by exercising the model? In summary, perhaps the conflict expressed in Figure 8 best reflects the discovery of emergent properties from the system. Pursuing the questions further leads to a growth in understanding exemplified by the findings based on the information presented in Figures 13, 14 and 15.

While the results presented in Section 6 are for a relatively simple case and some variation has been dealt with parametrically, the model is structured to deal with significantly increased complexity through the integration of more detailed analysis. Possibilities include adding features to incorporate real as opposed to ideal characteristics of the Rankine cycle (e.g., pipe losses, pumping losses). The mechanical design and more detailed sizing of components could also be added as could higher order heat transfer models that address the time and material dependencies of conduction in the heat exchangers. Including design robustness considerations are also desirable.

In Section 1, it was indicated that the thermodynamically oriented example presented herein is anticipated to provide the foundation for a significant body of future work and doctoral study in the "Systems Realization Laboratory" at the University of Oklahoma. Therefore, to conclude, the possible directions to be taken are identified.

Managing (Organized) Complexity – Future Work

In this work, the main focus will be on model development to grow complexity and the physical realism of the system (e.g., dimensions and materials). This will facilitate more detailed and practically-useful design input for small scale "power" plant systems through simulation.

Given the existing Stage 1 model, Stage 2 will include expansion focused on heat source issues: representation of aspects of the heat exchanger (boiler). While the current goals are moisture and efficiency based, the intent is to also model economic considerations. The selection decisions for the working fluids will be developed in line with options for lower temperatures and pressures applications inherent in an organic Rankine cycle.

For the heat exchanger, the first steps taken will include the thermodynamic modelling to address conduction leading to the specification of the required geometry (e.g., length and diameter of inner and outer pipe and material choice). Possibilities beyond Stage 2 include similar work with other system components as alluded to in Section 3 ("left" and "right" sides and further Rankine cycle refinements).

Managing (Disorganized Complexity) Uncertainty – Future Work

In this work, the focus will be on developing the modelling with respect to managing uncertainty. The research plan includes using the expanded thermal model, adding robustness considerations to address uncertainty and exploring the solution space using an Archimedean formulation, with sensitivity and post solution analysis.

In this paper the feasible solution space of the example thermal system was explored using a preemptive representation. As the example system model complexity grows, greater conflict in the goals is also anticipated. The next step, as described as the third in Section 2.2, is to explore the solution space by moving between the extremes to generate deeper understanding of the tradeoffs. An Archimedean (weighted sum) formulation of the goals can be utilized for this purpose.

Sensitivity and post solution analysis can be performed on a system by changing the bounds, relaxing or adding constraints, finding the limit (bounds) of the parameters and changing the target input data to be documented for the designer. Use of primal and dual information from a linear model (as generated by the ALP algorithm) may also provide new insights in exploring such a complex system.

Acknowledgments. We thank the University of New South Wales, Australia, for the financial support provided to Warren Smith for him to spend a sabbatical year at the Systems Realization Laboratory at the University of Oklahoma, Norman. Jelana Milisavljevic acknowledges the financial support from NSF Eager 105268400. Maryam Sabeghi acknowledges the NSF Graduate Research Fellowship that funds her graduate studies. Janet K. Allen and Farrokh Mistree acknowledge the financial support that they received from the John and Mary Moore chair account and the LA Comp chair account, respectively.

References

1. Marston, M., Allen, J.K., and Mistree, F., *The Decision Support Problem Technique: Integrating Descriptive and Normative Approaches.* Engineering Valuation & Cost Analysis, Special Issue on Decision-Based Design: Status and Promise, 2000. **3**: p. 107-129.
2. Muster, D. and Mistree, F., *The Decision Support Problem Technique in Engineering Design.* The International Journal of Applied Engineering Education, 1988. **4**(1): p. 23-33.
3. Mistree, F., Smith, W.F., Bras, B., Allen, J.K., and Muster, D., *Decision-Based Design: A Contemporary Paradigm for Ship Design.* Transactions SNAME, 1990. **98**: p. 565-597.
4. Mistree, F., Smith, W.F., Kamal, S.Z., and Bras, B.A., *Designing Decisions: Axioms, Models and Marine Applications (Invited Lecture)*, in *Fourth International Marine Systems Design Conference (IMSDC91)*. 1991, Society of Naval Architects of Japan: Kobe, Japan. p. 1-24.

5. Mistree, F., Hughes, O.F., and Bras, B., *Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm*, in *Structural Optimisation: Status and Promise*, M.P. Kamat, Editor. 1992, AIAA: Washington, DC. p. 251 - 290.
6. Mistree, F., Smith, W.F., and Bras, B.A., *A Decision-Based Approach to Concurrent Engineering, Chapter 8*, in *Handbook of Concurrent Engineering*, H.R. Paresai and W. Sullivan, Editors. 1993, Chapman-Hall: New York. p. 127-158.
7. Reddy, R., Smith, W.F., Mistree, F., Bras, B.A., Chen, W., Malhotra, A., Badhrinath, K., Lautenschlager, U., Pakala, R., Vadde, S., and Patel, P., *DSIDES User Manual*. 1992, Systems Design Laboratory, Department of Mechanical Engineering, University of Houston.
8. Mistree, F., Muster, D., Srinivasan, S., and Mudali, S., *Design of Linkages: A Conceptual Exercise in Designing for Concept.* Mechanism and Machine Theory, 1990. **25**(3): p. 273-286.
9. Chen, W., Simpson, T.W., Allen, J.K., and Mistree, F., *Satisfying Ranged Sets of Design Requirements using Design Capability Indices as Metrics.* Engineering Optimization, 1999. **31**(5): p. 615-619.
10. Lewis, K., Smith, W.F., and Mistree, F., *Ranged Set of Top-Level Specifications for Complex Engineering Systems, Chapter 10*, in *Simultaneous Engineering: Methodologies and Applications*, U. Roy, J.M. Usher, and H.R. Parsaei, Editors. 1999, Gordon and Breach Science Publishers: New York. p. 279-303.
11. Allen, J.K., Seepersad, C.C., Mistree, F., Savannah, G.T., and Savannah, G., *A Survey of Robust Design with Applications to Multidisciplinary and Multiscale Systems.* J Mech Des, 2006. **128**(4): p. 832-843.
12. Chen, W., Allen, J.K., and Mistree, F., *A Robust Concept Exploration Method for Enhancing Productivity in Concurrent Systems Design.* Concurrent Engineering, 1997. **5**(3): p. 203-217.
13. Chen, W., Allen, J.K., Tsui, K.-L., and Mistree, F., *A Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors.* Journal of Mechanical Design, 1996. **118**(4): p. 478-485.
14. Seepersad, C.C., Allen, J.K., McDowell, D.L., and Mistree, F., *Robust Design of Cellular Materials with Topological and Dimensional Imperfections.* Journal of Mechanical Design, 2006. **128**(6): p. 1285-1297.
15. Simpson, T.W., Chen, W., Allen, J.K., and Mistree, F., *Use of the Robust Concept Exploration Method to Facilitate the Design of a Family of Products.* Simultaneous engineering: Methodologies and applications, 1999. **6**: p. 247-78.
16. Simpson, T.W., Maier, J.R., and Mistree, F., *Product Platform Design: Method and Application.* Research in Engineering Design, 2001. **13**(1): p. 2-22.
17. Simpson, T.W., Seepersad, C.C., and Mistree, F., *Balancing Commonality and Performance within the Concurrent Design of Multiple Products in a Product Family.* Concurrent Engineering, 2001. **9**(3): p. 177-190.

18. Choi, H., McDowell, D.L., Allen, J.K., Rosen, D., and Mistree, F., *An Inductive Design Exploration Method for Robust Multiscale Materials Design.* Journal of Mechanical Design, 2008. **130**(3): p. 031402.
19. Choi, H.-J., Mcdowell, D.L., Allen, J.K., and Mistree, F., *An Inductive Design Exploration Method for Hierarchical Systems Design Under Uncertainty.* Engineering Optimization, 2008. **40**(4): p. 287-307.
20. McDowell, D.L., Panchal, J., Choi, H.-J., Seepersad, C., Allen, J., and Mistree, F., *Integrated Design of Multiscale, Multifunctional Materials and Products*. 2009: Butterworth-Heinemann.
21. Panchal, J.H., Choi, H.-J., Allen, J.K., McDowell, D.L., and Mistree, F., *A Systems-Based Approach for Integrated Design of Materials, Products and Design Process Chains.* Journal of Computer-Aided Materials Design, 2007. **14**(1): p. 265-293.
22. Seepersad, C.C., Allen, J.K., McDowell, D.L., and Mistree, F., *Multifunctional Topology Design of Cellular Material Structures.* Journal of Mechanical Design, 2008. **130**(3): p. 031404.
23. Chen, W., Meher-Homji, C.B., and Mistree, F., *Compromise: an Effective Approach for Condition-Based Maintenance Management of Gas Turbines.* Engineering Optimization, 1994. **22**(3): p. 185-201.
24. Hernamdez, G. and Mistree, F., *Integrating Product Design and Manufacturing: a Game Theoretic Approach.* Engineering Optimization+ A35, 2000. **32**(6): p. 749-775.
25. Koch, P., Barlow, A., Allen, J., and Mistree, F., *Facilitating Concept Exploration for Configuring Turbine Propulsion Systems.* Journal of Mechanical Design, 1998. **120**(4): p. 702-706.
26. Sinha, A., Bera, N., Allen, J.K., Panchal, J.H., and Mistree, F., *Uncertainty Management in the Design of Multiscale Systems.* Journal of Mechanical Design, 2013. **135**(1): p. 011008.
27. Pahl, G., Beitz, W., Feldhusen, J., and Grote, K.H., *Engineering design, A Systematic Approach*. Third English ed. 2007: Springer.
28. Bascaran, E., Bannerot, R.B., and Mistree, F., *Hierarchical Selection Decision Support Problems in Conceptual Design*. Engineering Optimization, 1989. **14**(3): p. 207-238.
29. Aird, T.J. and Rice, J.R., *Systematic Search in High Dimensional Sets.* SIAM Journal on Numerical Analysis, 1977. **14**(2): p. 296-312.
30. Lemmon, E.W. and Huber, M.L., *Implementation of Pure Fluid and Natural Gas Standards: Reference Fluid Thermodynamic and Transport Properties Database (REFPROP)*. 2013, National Institute of Standards and Technology, NIST.
31. Smith, W.F. and Mistree, F., *Monotonicity and Goal Interaction Diagrams for the Compromise Decision Support Problem*, in *ASME DE, New York*, B.J. Gilmore, D. Hoeltzel, D. Dutta, and H. Eschenauer, Editors. 1994, ASME. p. 159-168.

Complexity: Definition and Reduction Techniques

Some Simple Thoughts on Complex Systems

Jon Wade[1] and Babak Heydari[1]

Abstract. Complexity can mean many things to many people. This paper presents an empirical, relativistic definition of complexity relating it to the system of interest, the behavior which is to be understood, and the capabilities of the viewer. A taxonomy of complexity is described based on this definition. Complexity and complication are compared and contrasted to provide some context for these definitions. Several methods of reducing or managing complexity are presented, namely abstraction, transformation, reduction and homogenization. Examples are given for each of these.

Keywords: System complexity, system complication, system complexity management

1 The Age of Complexity

"I think that the next century (21st) will be the century of complexity"
- Stephen Hawking

The 20th century was certainly a time when we witnessed technological advances on a number of fronts including agriculture, transportation, communication, computation, energy, medicine and the like. However, the 21st century is one of complexity in which the interaction between these technologies, human behavior and the forces of nature form new and evolving systems.

A number of systems trends have been driving the exponential increase in system complexity. The notable reasons for this are an increase in both the scale and scope of interconnectivity and the increased participatory role of human agents. The dramatic increase in Software and Networking has had the major impact on interconnectivity. No longer are interactions limited by physical connectivity as they are in electro-mechanical systems. It is not possible to clearly define the impact of changes in software as was done in electro-mechanical systems. Hence, it is not as easy as it once was to determine what makes a car stop and go. Note that the cost of software in an automobile is greater than the cost of the steel.

Networking greatly impacts the quality of interconnectivity among agents of the same type. Notably, both the speed and richness of communication has increased with examples such as high-frequency trading and communication in the academic

[1] Stevens Institute of Technology, Castle Point on Hudson, Hoboken, New Jersey 07030, USA. jon.wade@stevens.edu, babek.heydari@stevens.edu

community. These factors also increase the quantity and connectivity between agents that previously had no connections. Socio-technical systems are driven by the human element. In the past, connectivity between humans was largely based on geography (according to one estimate the average American in the 1800's traveled an average of 50 meters per day [1].

Today, with the internet we are rapidly increasing the number of people who communicate and interact at a distance. In addition, through search technology we are discovering the shortest paths between two points. The hidden small world is now becoming much more visible. The six degrees of separation are being reduced, made accessible and turbocharged. The participation cost has been greatly reduced so that more people can participate which is an increase in scale, but is also an increase in scope as those who were once bystanders are becoming active participants as with Web 2.0. Finally, the impact of human behavior has risen to the degree that affects nature on a global scale. The world has truly become interconnected. The notion of systems with fixed boundaries, with agents playing fixed roles is quickly disappearing.

Complexity is the challenge, but what is complexity? How do we define it?

2 Complexity and Complication: Definitions

While many use the terms complexity and complication interchangeably, they have very different meanings. The literature follows these three basic categories of defining complexity.

Behavioral Definitions: The system is viewed as a black-box and the measures of complexity are given based on the outputs of the system. Behavioral measures include complexity as entropy in which the Shannon entropy of an output message from the system is regarded as a relatively objective measure of complexity [2]. Another definition is the effective complexity of the system in which the output of the system is divided into two parts: regularities and randomness. The effective complexity is the information content of the regularities whose determination is subjective and context dependent [3,4]. Statistical complexity defines complexity as the minimum amount of information from the past outputs of the system necessary to predict the future outputs [5]. This approach is problematic with respect to contexts which involve non-linear state changes to the system.

Structural Definitions: A measure or definition of complexity is given based on the structure/ architecture of a system. Many refer to the complexity of a system based solely on size. This is an objective definition that is perhaps the easiest to quantify. While complex systems quite often have a large number of components, complexity is more about how these components interact and are organized. For example, there are 45K protein coding genes in rice and 25K in Homo sapiens, but few would argue that rice is the more complex of the two. Another approach is to look at fractal dimensions [6]. While this definition is insightful, it is limited to certain types of structures. There are also hierarchical measures [7]. Simon claims that all complex systems have some degree of hierarchy and making building blocks on various levels is an im-

portant way that nature creates a complex system. However, the determination of the building blocks is arbitrary and context dependent.

Constructive Definitions: The complexity of the system is determined by the difficulty in determining its future outputs. The logical depth approach [8] shows how difficult it is to construct an object and regards the difficulty from a computational perspective, translating complexity into the number of steps needed to program a Turing machine to produce the desired output. This is a computational approach in which everything in the system needs to be digitized. Another approach is using thermodynamic depth [9] which is a more general form of logical depth which measures the amount of thermodynamic and informational resources necessary to construct an object. This method attempts to mimic the structure of a system by regenerating the output, but as this approach views the system as a black-box it can result in an unnecessarily large depth for the system.

There are also more general definitions of complexity [10, 11]. While all of these approaches have their merit, they do not seem to answer the essential question of what we mean when we use the word *complexity*.

The word *complicated* is from the Latin *com: together, plicare: to fold*. The adjective meaning of 'difficult to unravel' was first used in 1656 [12]. Interactions in complicated systems are often restricted with respect to interconnection, and can often be unfolded into simpler structures. In this case, decomposition works, while complex systems cannot be so easily unwoven. Complexity is related to the structure of the system.

Complexity is from the Latin *com: together*, *plectere: to weave*. The adjective meaning of 'not easily analyzed' was first recorded in 1715 [13]. Thus, from its first usage, complexity was synonymous with the ease of understanding something. The essence of complexity is interdependence. Interdependence implies that reduction by decomposition can't work, because the behavior or each component depends on the behaviors of the others.

Reductionism which alters the structure of a system cannot be used effectively as an analytic tool for a system whose behavior is critically dependent on these details. The structure often defines the system.

One can imagine complicated systems which are not complex, and complex systems which are not complicated. Figure 1 shows some examples of the possible permutations. The low complexity, low complication quadrant is populated with relatively simple inanimate objects, generally of a mechanical design. Systems engineering has traditionally been most successful in the high complication/low complexity quadrant, and system science in the low complication/high complexity quadrant. However, due to the need to engineer increasingly complex systems such as Systems of Systems and Socio-Technical systems, it is necessary to move systems engineering capabilities from the high complication/ low complexity quadrant, up to the high complication/ high complexity one.

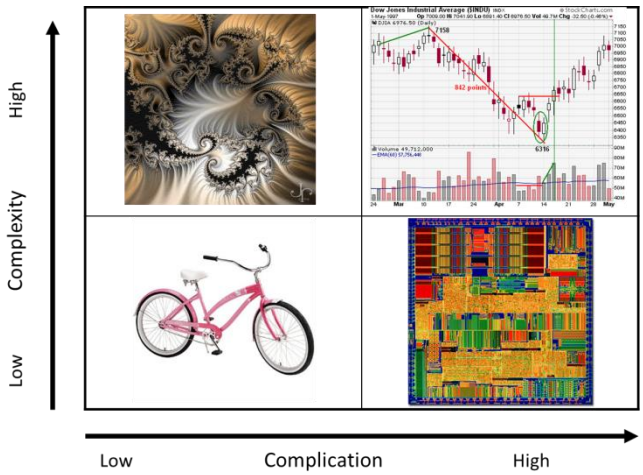


Fig. 1 Complication and Complexity: a) Mandelbrot Set, b) stock market, c) bicycle, d) processor chip.

Kurtz and Snowden [14], in the formulation of the Cynefin Framework, divide the decision making space into four domains, as shown in Figure 2. Roughly speaking, the "known" and "knowable" domains translate into the low-complexity, low-complication quadrant and the low-complexity high-complication quadrants, respectively, while the "complex" and "chaotic" domains are reflected in the high-complexity half of Figure 1.

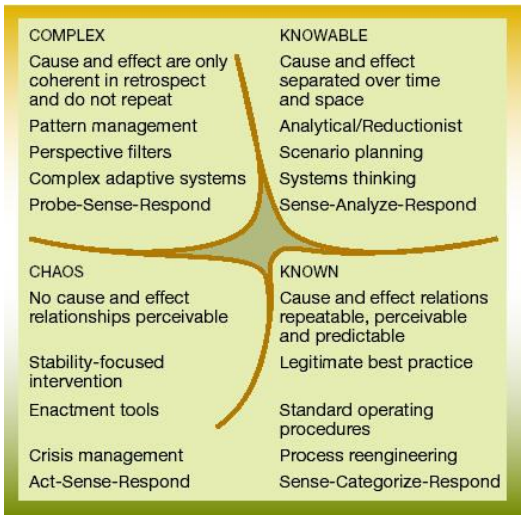


Fig. 2 Cynefin Framework (source: Kurtz & Snowden [14])

One might say that complexity is the degree of difficulty in understanding how a system works and thus how it behaves, but this might be too strong of a statement. With systems that are constantly evolving, it may not be possible to understand all the elements in a system, let alone how they interact, but it might be possible to predict how the system behaves. If we are to embrace complexity, then we need to accept the fact that understanding exactly how a system works may not be possible and we should focus on trying to understand how a system behaves. Thus, *embracing complexity involves a shift of emphasis from how something works to how it behaves.* This is major paradigm shift.

So what are the elements that make the behavior of a system difficult to predict?

One could ask the same question about something else which seems to be just as nebulous, such as 'beauty'. This is just as difficult to define and there probably isn't consensus on examples of beauty, let alone a consensus on the properties of an object, phenomenon or idea that imbues something with beauty. Just how do we objectively measure beauty? What are the common traits between things that are beautiful? Perhaps the same is true about complexity to some degree.

This is particularly difficult if one assumes that beauty is an intrinsic property independent of context and the observer. Rather than try to define beauty in terms of the characteristics of the object, perhaps it would make more sense to define it in terms of the effect that it has on the system which includes the observer. Such a definition might be, "beauty is something that brings pleasure to the observer." With this definition, it is clear that the beauty is dependent on the observer and context and one could imagine the means of perhaps measuring it through an electroencephalogram (EEG) or some other such device.

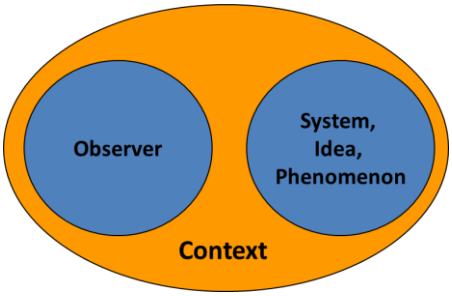


Fig. 3 Complexity – Relationship between Observer, System and Context.

Many others have discussed the critical importance of the observer on the system of interest. For example, philosophers such as John R. Searle [15] have divided the world into the ontologically objective and subjective; and into the epistemically objective and subjective. The Soft Systems Methodology (SSM) proposed by Checkland is a systematic approach [16] that is the result of continuing action research that

is used to analyze real-world problems by treating systems as epistemological rather than ontological entities, thus being dependent on human understanding. This is particularly important in the case of complex systems in which the analysis lacks a formal problem definition. This view is supported by the constructionistic epistemologists (first used by Jean Piaget [17]) who maintain that natural science consists of mental constructs that are developed with the aim of explaining sensory experience (or measurements) of the natural world. Some contributors to this philosophy include: dialectic constructivism (Piaget [18]), radical constructivism (Glaserfeld [19]) (Watzlawick [20]), (von Foerster [21]), (Bateson [22]), and Projective Constructivist Epistemology (Le Moigne [23]).

Taking the same approach that we took with 'beauty', as shown in Figure 3, 'complexity' may be defined as a relationship to the observer and context as:

"the degree of difficulty in accurately predicting the behavior of a system over time."

Thus, the degree of complexity is not only related to the system, idea or phenomenon of interest, but also is dependent on the context, the behavior in which the observer is interested, and the capabilities of the observer. Thus, there are a number of means by which the complexity of the system can be reduced without changing the system itself.

We appear to have these same issues with the definitions of other key terms in systems including such things as what is a "system" and the "-ilities" such as security, availability, flexibility, adaptability, etc. To avoid confusion, one should remember that the notion of 'systems' is a model that is employed to make sense of reality and the context and observer are all critical to this model building. Certainly the phrase 'system of interest' makes this point explicit.

Context is a critical aspect in the analysis of systems and is often neglected when discussing complexity. Context has three distinct faces, as described below, and the term is often used to refer to one or a combination of them depending on the situation [24].

Computational Context: When analyzing systems in the space-time domain, the initial and boundary conditions are quite important in determining the state of the system. Context in this sense can change by moving the boundaries of the system or changing the time reference.

Interpretative Context: As an observer, one can have different interpretations of the state of a system, based on the perceptual frame work s/he is using. The state of the system (or parts of it) can be interpreted as order/ signal or disorder / noise depending on the view point of the observer. A particular shape of a termite mound could be viewed as a magnificent structure, if the observer has seen a castle or some similar structure before. Otherwise, the shape can be completely meaningless.

Paradigmatic context: In some complex systems, especially those with human elements, a notion of context emerges as a result of the combination of the internal states of the agents and their interactions. This notion of context includes a set of

rules, standards, collective perceptual framework or a value structure. This can be thought of as a generalization of what Thomas Kuhn calls “paradigm” [25] specifically for the scientific community. This is also aligned with the notion of "socially constructed phenomena" that we have already talked about.

In most of the discussions about the context of a system, people refer to the first and sometimes the second form, but rarely the third. The important point is that in human-centric complex systems, there is a cyclical causation between the last two forms of context. In a way, the paradigmatic form shapes the internal interpretative context which itself influences the paradigm of the system.

3 Factors of Complexity

Complexity is far too, well complex, to be described with a scalar quantity. Rather there are several dimensions which reflect the overall difficulty in accurately predicting the future behavior of a system. The following are a set of factors that relate to the overall system of observer, context and system which is consistent with the definition of complexity that we have established. These factors consists of two major components. The first relates to desired accuracy and scope of the prediction and the second relates to the degree of difficulty in obtaining the desired predictive capability.

Prediction quality can be determined to depend upon the achievable precision, timescale and breadth of context. The following are some of the ranges for each of these which are relative to the system of interest.
The precision of predictive capability ranges from:

- Exact (approximate) state is deterministic
- Exact (approximate) states have stochastic probabilities
- Exact (approximate) states have stochastic ordering
- Future (current) states are ill-defined
- Future (current) states are largely unknown

The timescale of predictive capability ranges from:

- Beyond the expected life of the system
- Accepted life of system
- Significant fraction of life of system
- Small fraction of life of system
- Only for small deviations from current state

The breadth of context for the predictive capability ranges from:

- All imaginable contexts
- All likely contexts
- Some contexts
- Only current context

The desired quality level of prediction can be created by specifying a vector in this space.

Prediction difficulty is determined by three critical factors. The first factor is the degree of difficulty in understanding the relationships that govern the interactions and behaviors of the components. The second factor is the degree of difficulty in knowing the current state of the system to the level necessary to apply the relationship knowledge. The final factor is the degree of difficulty in knowing or computing the behavior of system. One of the most challenging aspects of this computation is ability to discover and predict unforeseen emergent behaviors. Quite often these emergent behaviors are dependent upon relationships that are not well understood and may be critically dependent on the system's initial conditions. The following are some of the ranges for each of these which are each relative to the system of interest.

The difficulty in understanding relationships governing interactions and behaviors:

- Essential relationships are well understood quantitatively
- Essential relationships are well understood qualitatively
- Essential relationships are not well understood
- It is unknown which are the essential relationships

The difficulty in acquiring necessary information needed to make a prediction:

- Essential information is known
- Essential information may be acquired with significant effort
- Essential information may not be acquired in that it is not measurable or the act of measuring it causes it to substantially change
- It is unknown what constitutes essential information in the future (currently)

The difficulty in computing the behavior of the system:

- Behavior of the system is evident through mental analysis
- Behavior of the system may be calculated in the desired time on a personal computer
- Behavior of the system may be calculated in the desired time on a super computer (1000x PC)[2]
- Behavior of the system may be calculated in the desired time on a foreseeable super computer (1Million x PC)
- Behavior of the system may be calculated on a theoretical quantum computing system
- Behavior of the system may not be calculable

For example, the relationship of factors is fairly well known in a weather system, but the challenge is to understand the current state to the necessary level of accuracy and being able to calculate the resultant weather more quickly than the actual phe-

[2] A factor of 1000x in computing is approximately equal to 15-20 years into the future.

nomenon. Climate change is much more difficult as the relationships between the relevant factors are not well understood.

This approach embraces complexity in that the taxonomy is not based on how the system works, but rather how it behaves. While other taxonomies may be used to describe the physical characteristics of the system, this may lead to erroneous conclusions about the systems complexity per our definition. For example, a simple cellular automata system may be composed of few agents, have well defined communication and simple rules for behavior, yet result in behaviors that are very difficult to predict. The converse is true as well.

4 Complexity Reduction

What can be done to reduce complexity, that is, to make system behavior more predictable? While some such as A. Berthoz [26] have proposed a set of organizing principles based on biological systems for "simplexity", the means to provide complementary relationships between simplicity and complexity, this paper is intended to describe approaches by which to reduce the difficulty in predicting the possible future behaviors of systems. Four possible approaches described in this paper to reduce complexity are: reduction, homogenization, abstraction, and transformation, each of which is described below.

4.1 Reduction

Reduction is the process of removing superfluous elements from the system, either in practice or in implementation, and/or limiting the context under which the system is allowed to operate and reducing the state space to something which is understood. For example, when using a subway system, most riders are interested in how to travel from point A to point B, making the necessary connections. A map, as shown in Figure 4, provides just this amount of information, by eliminating elements that are not relevant to understanding this particular behavior. It should be noted that reductionism in this case does not eliminate structure, but rather makes the essential structure much more visible.

Reduction in context can be used when a system is moving into a regime in which its operation is not valid, such that steps are taken to move it back into a known space. For example, an integrated circuit's operation is well understood within certain temperature, voltage and frequency constraints and it is not allowed to operate outside this regime where it becomes far less predictable and perhaps chaotic. Thus, a potentially complex system is transformed into one that while being complicated is highly predictable.

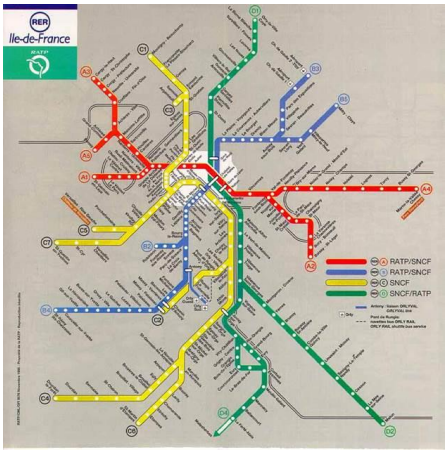


Fig. 4 Paris Metro Map (Source: http://www.ionbee.net/media/parismetromap.jpg)

4.2 Homogenization

Homogenization is somewhat related to reduction in that it provides the possibility to reduce the types of elements or agents by classifying them into sets that are relatively indistinguishable or homogeneous. This is the technique that allows statistics to be applied to situations rather than being forced to understand the behavior of each element. For example, it would be intractable to predict the behavior of more than a few molecules of air, yet the aggregate behavior of 10^{27} such molecules, namely pressure, volume and temperature, can be predicted with a simple ideal gas model if each molecule is treated as being indistinguishable. One should remember that if the behavior of interest is that of the individual molecules, then the system is highly unpredictable, and highly complex. Hence, the same system can be highly complex or very simple depending on the type of behavior of interest and the context of operation.

One must be very careful when applying the technique of homogenization not to overly simplify the model of the system to the point where it is not useful in predicting the desired behavior. For example, one part in a billion can make a big difference in certain reactions. In semiconductors doping levels on the order of 1 part per 100,000 can increase the conductivity of a device by a factor of 10,000 times. There are many systems in which a small amount of inhomogeneity can create starkly different behaviors. For example, pure water in isolation at 1 atmosphere pressure will freeze at -42 degC or even as low as -108degC if cooled sufficiently quickly, while water in the presence of dust or other impurities that can serve as crystallization sites freezes at the familiar 0 degC.

4.3 Abstraction

Abstraction is essentially the ability to decouple elements in a system and transform it from a woven to a folded statement in which interactions are restricted. A good ex-

ample for this part is language and thought: the more abstraction we enter in our language by encapsulating a notion into a word, the more we will be able to deal with the complexities of a conceptual problem. In fact, the creation of jargon in a scientific field, is a form of abstraction that serves to reduce the complexity of that field. Mead and Conway's book, *Introduction to VLSI Systems*, published in 1980 [27] codified this layering, as shown in Figure 5, and helped to transform complexity to complication in VLSI systems. This success has allowed the creation of incredibly complicated systems with deterministic behavior which has driven software complexity and networking which has driven us to very complex systems.

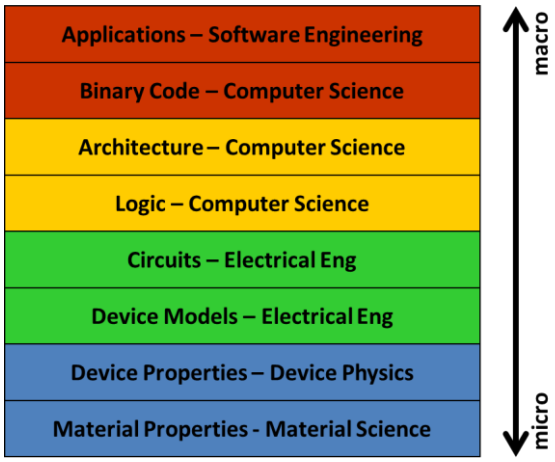


Fig. 5 Layering within Computing Systems utilizing VLSI Technology

It is also interesting to note that abstraction reduces the complexity at the existing boundary of a system, but it also creates a new level of complexity. In fact, this is one of the main mechanisms behind the progress of various fields in human knowledge: Efforts to reduce complexity results in creation of new level of abstractions. The resulting abstractions create a new boundary for the system and generate a new form of complexity, and the cycle continues.

4.4 Transformation

Transformation is a technique in which the problem space is altered such that it becomes more tractable and predictable. An example of this is taking a system that is very difficult to understand in the time domain and performing analysis on it in the frequency domain. Moving from systems governed by rules to ones governed by principles may be seen as a form of transformation. Sometimes perspective can have an enormous impact on one's ability to understand a system's behavior.

One of the important studies in systems science is that of networks. In this case, the system is analyzed with a transformation of its precise structure, to one that is

characterized by local and non-local connectivity and diameter (degrees of separation). This transformation enables a significant reduction in the number of factors that need to be addressed to understand the behavior of the system. Each of the systems shown in Figure 6 is composed of networks of systems that experienced evolutionary processes and as a result have a similar network structure with respect to connectivity and diameter. In this case these are composed of 'scale free' networks whose degree distribution follows a power law, such that a small number of nodes have a large number of interconnections, while most have a small number of interconnects.

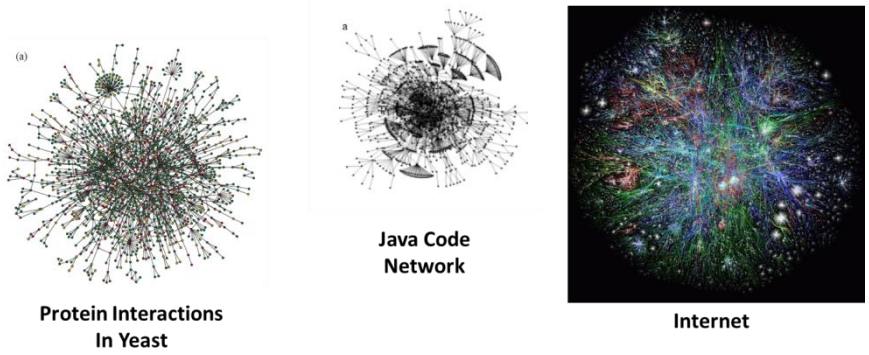


Fig. 6 Network Structures in Evolving Systems

It is known that these types of systems are rather resilient to random faults or attacks, yet are very susceptible to failure in the "too big to fail" nodes. These systems also involve tipping points which when tipped places the system in a different state such that it is usually not easy to return to the prior state. Thus, much can be understood about the system based on a small amount of information.

5 Conclusion

In summary, the following are some of the significant points made in this paper. First, system complexity is increasing exponentially due to increases in both the scale and scope of interconnectivity and the role of human agents in the system. Embracing complexity requires a paradigm shift from attempting to deterministically understand how a system works to how a system stochastically behaves. While one should not give up on understanding the inner-workings of a system, it cannot be assumed that complete knowledge of the system will be possible.

Complexity can defined as: "the degree of difficulty in accurately predicting the behavior of a system over time." This definition includes the critical framework of the system, observer and context. Thus, the complexity of a system can be simultaneously very high or very low depending of the type of behavior that the observer is trying to predict. Complicated systems may have many parts, but the scope and be-

havior of these interactions are generally well constrained, and their behavior is deterministic. Complex systems, on the other hand, have a much richer set of interactions, and have behaviors that are impossible to accurately predict. System complexity can be viewed from a multi-dimensional taxonomy including precision of prediction, time scale of prediction, difficulty in acquiring necessary information, and breadth of context.

Complexity can be reduced through reduction, homogenization, abstraction and transformation. A final general note to make, which seems obvious, is that when using any of these techniques, some information about the system is lost. Whether that piece of information is crucial or superfluous depends on the context and that particular application of the system. It is always essential to have the assumptions behind each of these four techniques in mind. Many systems failures are the result of a particular simplification technique being used successfully in one context and then being misapplied in another context in which the missing information is critical.

The challenge of science as Einstein put it, is to make things "as simple as possible, but no simpler."

References

1. Urry, John. 2007. *Mobilities.* Cambridge, Polity.
2. Shannon, C. E. (2001). A mathematical theory of communication. ACM SIGMOBILE Mobile Computing and Communications Review, 5(1), 3-55.
3. Gell-Mann, M., & Lloyd, S. (1996). Information measures, effective complexity, and total information. Complexity, 2(1), 44-52.
4. Heydari, B., & Dalili, K. (2012). Optimal System's Complexity, An Architecture Perspective. Procedia Computer Science, 12, 63-68.
5. Feldman, D., Crutchfield, J. (1998) Measures of statistical complexity: Why?, Physics Letters A, Volume 238, Issues 4-5, February 1998,
6. Mandelbrot, B. B. (1977). Fractals: form, change and dimension. San Francisko: WH Freemann and Company.
7. Simon, H. A. (1962). The architecture of complexity. In Proceedings of the American Philosophical Society.
8. Bennett, C. H. (1995). Logical depth and physical complexity. In The Universal Turing Machine A Half-Century Survey (pp. 207-235). Springer Vienna.
9. Bennett, C. H. (1995). Logical depth and physical complexity. In The Universal Turing Machine A Half-Century Survey (pp. 207-235). Springer Vienna.
10. Mitchell, M. (2009). *Complexity: A guided tour*. Oxford University Press.
11. Horgan, J. (1995). From complexity to perplexity. *Scientific American*, *272*(6), 104-109.
12. complicated. Dictionary.com. *Online Etymology Dictionary*. Douglas Harper, Historian. `http://dictionary.reference.com/browse/complicated` (accessed: April 1, 2014).
13. complex., *ibid.*
14. Kurtz, C.& Snowden, D. (2003). The new dynamics of strategy: Sense-making in a complex and complicated world. IBM Systems Journal Vol 42 No 3, 2003.
15. Searle, J. R. (1990). The Mystery of Consciousness, New York Review Books.

16. Checkland, Checkland, Peter B. & Poulter, J. (2006) Learning for Action: A short definitive account of Soft Systems Methodology and its use for Practitioners, teachers and Students, Wiley, Chichester.
17. Piaget, J. (1967). Logique et connaissance scientific [Logic and scientific knowledge]. Paris: Gallimard.
18. Piaget, J. (1970). L'épistémologie génétique. PUF, Paris.
19. Glaserfeld, von E. (1995), Radical Constructivism. A way of knowing and learning. The Falmer Press, London.
20. Watzlawick,P. (1977). How real is real? New York, Vintage.
21. Foerster, von H. (1984) Observing systems. Intersystems Publications Seaside, C.A. 2nd ed.
22. Bateson, G. (1972). Steps to an ecology of mind. Ballentine Book., New York.
23. Le Moigne, J.L. (1995). Que sais – je?, Les épistémologies constructivistes. PUF, Paris.
24. Heydari, B., Mitola, J., & Mostashari, A. (2011, April). Cognitive context modeling in the socio-technical systems. In Systems Conference (SysCon), 2011 IEEE International (pp. 272-277). IEEE.
25. Kuhn, T. (1962). The Structure of Scientific Revolutions. The University of Chicago Press, Chicago.
26. Berthoz, A. (2012). Simplexity: Simplifying Principles for a Complex World. Yale University Press.
27. Mead, C., & Conway, L. (1980). *Introduction to VLSI systems* (Vol. 1080). Reading, MA: Addison-Wesley.

Requirements for Single Pilot Operations in Commercial Aviation: A First High-Level Cognitive Function Analysis

Guy André Boy

Human-Centered Design Institute, Florida Institute of Technology
150 West University Boulevard, Melbourne, FL 32901, USA
gboy@fit@fit.edu

Abstract. Aeronautical engineering never stopped decreasing the number of technical crewmembers in commercial aircraft since the 1950s. Today, a new challenge has to be taken: single pilot operations (SPO). SPO consist of flying a commercial aircraft with only one technical crewmember in the cockpit, assisted by advanced onboard systems and ground operators providing flying support services. This next move is motivated by cost reduction, and must satisfy the same or better level of safety currently guaranteed with two-crewmen cockpit. This is a human-centered design (HCD) problem where decision-makers have to take risks. This paper presents an approach to risk taking in systems engineering. This approach is illustrated by the presentation of the difficult problem of SPO HCD, and the underlying function allocation problem.

1 Introduction

This paper is strongly based on the experience of the author in the analysis, design and evaluation of aeronautical systems, mainly cockpit systems, and more specifically, the shift from three to two crewmen cockpits in commercial aircraft in the beginning of the eighties (Boy, 1983; Boy & Tessier, 1983, 1985). Task analysis and multi-agent modeling and simulation supported this work. The MESSAGE[1] model was developed to represent and better understand interactions among various human and machine agents, such as aircrew members, aircraft systems and air traffic control (ATC). A series of indicators were developed to assess workload in particular. These indicators were tested both in simulations and in real flights, and were actually used during aircraft certification campaigns. They measured both physical ergonomics and cognitive variables. One of the main results of the MESSAGE project was the development of a new approach to function analysis that could support investigations in multi-agent work environments. When the number of crewmembers changes, there is necessarily a new distribution of functions (i.e., roles and jobs) and tasks. In addition, teamwork also changes. We then need to redefine the various functions and interac-

[1] Modèle d'Equipage et Sous-Systèmes Avion for la Gestion des Equipements (Model of aircrew and aircraft sub-systems management).

tions among agents implementing these functions. This is what the Cognitive Function Analysis (CFA) enables us to do (Boy, 1998, 2011). An agent's cognitive function is defined by its role, context of validity and a set of resources that enable the agent to satisfy her/his/its role. CFA enable the generation of cognitive function networks superimposed on multi-agent networks, and improves our understanding of appropriate function allocation.

Today, motivated by cost reduction, the shift from two pilot operations to single pilot operations (SPO) requires us to investigate how cognitive functions will be redistributed among humans and systems. We put "humans" plural because even if the objective is to have a single pilot in the cockpit, there will be other human agents on the ground or onboard (e.g., flight planners, flight followers, and flight attendants) who could be involved. This function allocation process is typically done using CFA to design the first prototypes and prepare human-in-the-loop simulations (HITLS), and after HITLS to refine the definitions of the various cognitive functions involved and their inter-relations (Boy, 2011). In addition, HITLS enable us to discover **emerging cognitive functions** (ECF), which cannot be deliberately defined in the first place. ECF can only be discovered at use time. Consequently, this approach imposes a new challenge in systems engineering that is to articulate CFAs and HITLS. Risks in the choice of configurations (i.e., cognitive functions of the agents involved) and scenarios (i.e., tasks and chronologies of events) are mitigated by Subjects Matter Experts (SMEs). This approach enables us to eliminate unsatisfactory solutions from the very beginning of the life cycle of a product. We are not working on short-term predictions but on tests of possible longer-term solutions. The whole challenge is in creativity, mandatory for the generation of these possible solutions. Creativity in human-systems integration is typically the product of experienced design thinking and incremental expertise-based syntheses.

2 What is Cognitive Function Analysis (CFA)?

CFA can be used to both analyze current multi-agent interactions, and future possible scenarios and configurations in two orthogonal spaces: the resource space and the context space. The resource space includes logical networks of human and system functions. The context space includes relevant situations embedded in progressively generic context patterns. For example, when we want to represent a function of responsibility delegation from a human to a system, we represent the various resources that both human and system require to support it, and the various context levels in which its resources can be used. There may also be embedded cognitive functions (i.e., cognitive functions of cognitive functions). As a whole, this approach enables us to study the intrinsic complexity of the generated resulting cognitive function network. Use of CFA methodology acknowledges the intrinsic complexity involved in multi-agent socio-technical systems and offers a path to systematically analyze component interactions that give rise to unanticipated emergent behaviors, attributes, and properties. We have used this approach to study and incrementally redesign automation in commercial aircraft cockpits (Boy, 1998; Boy & Ferro, 2003).

At the moment, commercial aircraft cockpits include two crewmembers, a pilot flying (PF) and a pilot not flying (PNF) – also called pilot monitoring (PM). Typically, the PF is in charge of the control of the aircraft, and the PNF is in charge of system monitoring, communication with the ground, and safety monitoring of flight progress. When agent roles and number change within an organization (i.e., when the cognitive function network changes), there is a re-distribution of the various authorities. Authority is about control (i.e., being in charge of something) and accountability (i.e., you need to report to someone else). CFA enables us to study authority re-distribution by making explicit the various roles, contexts and resources, and the links among them. When we moved from three to two crewmembers in cockpits, we needed to study the re-distribution of cognitive functions between the two crewmembers and the new systems (highly automated) that were executing tasks that the previous third crewmember was executing in the past. The main problem was to identify the emerging cognitive functions induced by the new human-system-integration (Boy & Narkevicius, 2013). Pilots were moving from classical control tasks to systems management tasks. For that matter they had to create and learn new cognitive functions to accomplish the overall flying task.

The major advantage of two crewmen cockpits is redundancy (i.e., it is better to have two pairs of eyes and two brains, than only one of each). Safety deals with stability, resilience and therefore cognitive redundancy. This is something that will need to be challenged and tested in the SPO framework. In particular, a comparison of the current two crewmen cockpit operations with SPO should also be conducted. When we shifted from three to two crewmen cockpits, we first developed a time line analysis (TLA), which consists in developing scenarios of events as well as interactions among the various agents involved (e.g., captain, first officer, ATC, aircraft). Since then we made lots of progress in usability engineering and TLA could be combined with cognitive function analyses. We also ran simulations that enabled to play these scenarios and observe activities of the various agents.

As a general standpoint, commercial airline pilots are typically involved as subject matter experts (SMEs). The various variables and processes that we typically study are the followings: pilot's goals, workload (or task-load during the TLA), human errors (i.e., possible error commissions and recovery processes), situation awareness, decision-making process, and action taking. Scenario data are chronologically displayed on a classical spreadsheet, which can be upgraded as needed when the analysis progresses. An example of such an approach is provided in (Boy & Ferro, 2003). Once this first task-based CFA is done, we play the same scenarios (or updated scenarios – we always learn new things during a CFA, then we can exploit the findings to upgrade original scenarios) on cockpit simulators with SMEs. The main goal of this research phase is to discover emerging cognitive functions; this is the main advantage of using HITLS. In classical function allocation methods (Fitts, 1951), we do not see emerging cognitive functions because they are used a priori and not incrementally using HITL simulation results. When we deal with change management, these emerging cognitive functions are tremendously important to discover as early as possible to avoid potential catastrophic surprises later on (Boy, 2013).

3 Stating the SPO problem: Evolution or Revolution?

The number of aircrew in cockpits was reduced over the years during the last 60 years or so going from 5 until the 1950s when the Radio Navigator was removed (the radio navigator was dedicated to voice communication equipment), to 4 until the 1970s when the Navigator was removed (when inertial navigation systems were introduced), to 3 until the 1980s when the Flight Engineer was removed (new monitoring equipment for engines and aircraft systems were introduced), to 2 until now. Two-aircrew cockpits have been the standard for three decades. This progressive elimination of technical crewmembers in commercial aircraft cockpits results from the replacement of human functions by systems functions. These functions are both cognitive and physical. The reason we only talk about cognitive functions is because electronics and software progressively dominated the development of systems. Today, it is clear that many onboard systems have their own cognitive functions in terms of role, context of validity and resources used (Boy, 1998).

Current technology indicates that we can move to single pilot operations (SPO). Two institutions support this new shift: NASA in the US and ACROSS[2] in Europe. It is clear that the main goal of moving from two-crewmen cockpit operations to single pilot operations (SPO) is the reduction of costs. We now need to investigate how safety would be impacted by this shift. It is true that SPO is already well experienced in general aviation (GA); in this case, we know its advantages and drawbacks. In particular, ATC is already familiar with interaction with single pilots. In addition, military fighters are operated with only one pilot in the cockpit.

We foresee two main approaches to SPO. The former is an **evolutionary** approach that continues the move from 5 to 4 to 3 to 2 to 1 where automation is incrementally added as the aircrew number is reduced. The main issue is pilot incapacitation. We always certify an aircraft entirely safe for (n-1) capacitated flying pilot(s). When n=1, there is a discontinuity, and the piloted aircraft becomes a drone. We then need to define ground support and/or flight attendant support. The latter is a **revolutionary** approach that breaks automation continuity and goes to the design of a fully automatic flying machine (commonly called a drone or a flying robot). The problem becomes defining human operator's role. Consequently, human-robot interaction activity needs to be entirely defined from the start within a multi-agent environment, and not only when the pilot is incapacitated, having a single agent approach in mind.

In both approaches, function allocation is a major mandatory endeavor. In the evolutionary aircrew-reduction approach, it is purposeful to compare the differences and commonalities between general aviation (GA) single-pilot resource management (SRM) and commercial aviation SPO SRM (to be defined). The FAA has identified 6 tenets of SRM in GA[3]: task management; risk management; automation management; aeronautical decision-making; control flight into terrain awareness; and situation awareness. Another important question is the definition of the role (job) of the single pilot in SPO and related operations support (i.e., procedures, automation, and problem solving skills). It is also crucial to find out risks involved in SPO as early as possible

[2] ACROS: Advanced Cockpit for Reduction Of Stress Consortium.

[3] FAA Order 8900.2, General Aviation Airman Designee Handbook http://fsims.faa.gov/wdocs/orders/8900_2.htm

before delivery. This is why fast-time simulations and human-in-the-loop simulations are planned and carried out from the beginning of the design process. Finally, it is important to identify and design a new cockpit configuration for SPO integrated into a global infrastructure covering the entire air traffic management (ATM).

In the revolutionary approach, instead of looking for what we loose when we remove the first officer (a negative approach where "overload" is studied and cumulative assistance is searched), it is urgent to understand what function allocation should be developed between the SPO aircrew and systems that will need to be developed (a positive approach where situation awareness, decision making and human-machine cooperation are studied and developed from the start). What will be the role of an aircrew flying a drone? There is a major difference between controlling and managing a transport drone from the ground and inside it. The latter is likely to be more socially accepted by passengers. Therefore, the primary question is the definition the role/job of this new type of aircrew; use of socio-cognitive models and complexity analyses will be necessary. In addition, we need to find out emerging human factors issues such as situation awareness, decision-making (who is in charge and when), fatigue, and incapacitation. This should be studied in nominal and off-nominal situations.
The major distinction between RPAS (Remotely Piloted Aircraft Systems[4]) and SPO of drones is crucial. When the person responsible for safety, success and wealth of a mission (a flight) is himself/herself directly involved (life-critical embodiment instead of remote control), he/she will have totally different relationships with the machine being controlled and managed. This approach does not remove the need for ground assistance. There will be decision to make whether or not we want to make pilot's manual reversion possible, and/or have RPAS as an emergency/recovery possibility. In any case, board and ground personnel, organizations and technology roles should be defined in concert (i.e., consider complex and non-linear systems and design/test global solutions) and not in isolation as it is done today (i.e., simplify problems, linearize and find local solutions). Tests will be performed using various human factors metrics and methods including workload, skills, knowledge and performance assessments. Other metrics can be used such as simplicity, observability, controllability, redundancy, (socio-)cognitive stability, and cognitive support.

4 Cognitive Function Analysis of Single Pilot Operations

Using CFA to define SPO leads to the identification of cognitive functions for the various agents including the pilot (or another qualifier in the SPO context), ground operators and systems. Each of these agents has a set of cognitive functions providing him, her or it with some degree of authority. Authority can be viewed as control (i.e., the agent is in charge of doing something and control the situation) and accountability (i.e., the agent is accountable to someone else). Control can be either handled directly or delegated to other agents who have authority to execute well-defined tasks. In this latter case, these other agents (should) have appropriate and effective cognitive func-

[4] http://ec.europa.eu/enterprise/sectors/aerospace/uas/index_en.htm

tions and are accountable to the agent delegating. CFA enables to rationalize the allocation of cognitive functions among agents.

Technical crews (or pilots) have the role and authority of bringing a set of passengers from a location A to another location B. They have the primary responsibility for safety, efficiency and comfort of the passengers. In SPO, cognitive functions of current PF and PNF are distributed among technical crews, ground operators and new systems. In particular, technical crew cognitive functions have a set of resources distributed among ground operators and aircraft/ground systems. Ground operators have different cognitive functions that can be named dispatching, ATC coordination, crew scheduling, maintenance triggering, customer service, and weather forecast. All these cognitive functions can be supported by systems when they are well-understood and mature. Dispatching and piloting are currently associated to develop a flight plan, find out what fuel quantity pilots should take, meet weight and balance requirements, ensure compliance with the minimum equipment list (MEL), de-conflict with other aircraft, help in case of equipment failure and, more generally, guide the flight from gate to gate.

Normal piloting cognitive functions consist in reading checklists, cross-checking life-critical information, trouble-shooting and recovering from failures, fuel monitoring, and so on. Abnormal and emergency cognitive functions are triggered by specific conditions such as engine failure, cabin depressurization, fuel imbalance and so on. Current air traffic controllers have specific dispatch cognitive functions. Their job will change with SPO and will need piloting cognitive functions in the case of malfunctions in the airspace, including pilot incapacitation and its duality, total system failure. Consequently, they will need tools such as in aircraft cockpits. The whole ATC workstation will evolve toward an ATM/piloting workstation for SPO. In addition, they will not have to control only one aircraft but, in some cases, several.

A first cognitive function analysis shows that there are functions that can be allocated to systems such as checklists-based verifications and crosschecking of life-critical information. As always, allocating functions to systems requires maturity verification. Whenever technology maturity is not guaranteed, people should be in charge and have capabilities guarantying good situation awareness. In any case, tests are mandatory. In the above-defined revolutionary approach to SPO, we typically think about lower levels of control being entirely automated (i.e., trajectory control and management); human agents only act on set-points for example. We need to be careful however that the SPO pilot will be aware of the crucial internal and external states of his/her aircraft environment. He/she will also need to be knowledgeable and skilled in aviation to perceive and understand what is going on during the flight as well as act on the right controls if necessary. Full automation does not remove domain knowledge and skills in life-critical systems.

Symmetrically, some aircraft systems (or artificial agents) should be able to monitor pilot's activity and health. This induces the definition, implementation and test of new kinds of system cognitive functions based on physiological and psychological variables. Obviously, this will require sensors that could be physiologically invasive (e.g., electro-encephalograms) or non-invasive (e.g., cameras). In any case, pilots will have to accept to be monitored. Pilot's activity and health monitoring can be done by aircraft systems and also by ground operators.

Other processes and technology that need to be human-centered designed and developed are related to collaborative work. In this new multi-agent world, agents have to collaborate and be supported for this collaboration. Computer-supported cooperative work (CSCW) technology and techniques need to be developed to this end. When an agent fails for example, whether a human or a system, recovery means and strategies should be in place to continue the flight safely. It is also important, in this increasingly automated multi-agent world, to keep enough flexibility. HITLS could be used to find out the effectivity of possible solutions.

5 Conclusion

This paper showed a major distinction between a classical evolutionary approach and a revolutionary approach for SPO. A first high-level cognitive function analysis (CFA) was carried out showing the contribution of experience and creativity leading to innovation.

Studying function allocation among people and systems from the beginning enables the development of socio-cognitive models, which further support human-in-the-loop simulations, and incrementally design systems, organizational setups and job descriptions in order to innovate in a human-centered way (e.g., define SPO). The TOP (Technology, Organization and People) model should always support the HCD process leading to technological, organizational and jobs/functions solutions.

This first high-level CFA needs to be further developed as SPO TOP solutions are incrementally developed. We need to discover human and technological weaknesses, and design appropriate redundancy in the form of technology, onboard personnel support and ground support. Studying multi-agent collaborative work (humans and systems), it is important to improve our understanding of authority and context sharing (distributed cognition), improve mutual feedback (cross-checking, cross-communication, intent recognition), as well as responsibility and accountability. We found that the Orchestra model (Boy, 2013) was a good framework to handle this kind of innovation in the aeronautical domain.

References

1. Boy, G.A. (1983). Le Système MESSAGE : Un Premier Pas vers l'Analyse Assistée par Ordinateur des Interactions Homme-Machine (The MESSAGE System: A first step towards computer-Assisted analysis of human-machine interactions). *Le Travail Humain Journal*, Tome 46, no. 2, Paris, France.
2. Boy, G.A. & C. Tessier (1983). MESSAGE : An Expert System for Crew Workload Assessment. *Proceedings of the 2nd Symposium of Aviation Psychology*, OHIO State University, USA.
3. Boy, G.A. & C. Tessier (1985). Cockpit Analysis and Assessment by the MESSAGE Methodology. *Proceedings of the 2nd IFAC/IFIP/IFORS/IEA Conf. on Analysis, Design and Evaluation of Man-Machine Systems*, Villa-Ponti, Italy, September 10-12. Pergamon Press, Oxford, pp. 73-79.

4. Boy, G.A. (1998). *Cognitive Function Analysis*. Greenwood/Ablex, CT, USA; ISBN 9781567503777.
5. Boy, G.A. & Ferro, D. (2003). Using Cognitive Function Analysis to Prevent Controlled Flight Into Terrain. Chapter of the *Human Factors and Flight Deck Design* Book. Don Harris (Ed.), Ashgate, UK.
6. Boy, G.A. (2011). Cognitive Function Analysis in the Design of Human and Machine Multi-Agent Systems. In G.A. Boy (Ed.) *Handbook of Human-Machine Interaction*. Ashgate, UK, pp. 189-206
7. Boy, G.A. (2013). *Orchestrating Human-Centered Design*. Springer, U.K. ISBN 978-1-4471-4338-3.
8. Boy, G.A. & Narkevicius, J. (2013). Unifying Human Centered Design and Systems Engineering for Human Systems Integration. In *Complex Systems Design and Management.* Aiguier, M., Boulanger, F., Krob, D. & Marchal, C. (Eds.), Springer, U.K. 2014. ISBN-13: 978-3-319-02811-8.
9. Fitts, P. M., (Ed.). (1951). *Human Engineering for an effective air-navigation and traffic-control system*. Columbus Ohio: Ohio State University Research Foundation.

Urban Lifecycle Management: System Architecture Applied to the Conception and Monitoring of Smart Cities

Claude Rochet [1], Florence Pinot de Villechenon [2]

[1] Professeur des universités
Aix Marseille Université
IMPGT AMU CERGAM EA 4225
ESCP Europe Paris
Claude.rochet@univ-amu.fr

[2] Directeur CERALE Centre d'Etudes et de
Recherche Amérique latine Europe
ESCP Europe Paris
pinot@escpeurope.eu

Abstract: At date, there is no standardized definition of what a smart city is, in spite many apply to propose a definition that fit with their offer, subsuming the whole of the city in one of its functions (smart grid, smart mobility…). Considering the smart cities as an ecosystem, that is to say a city that has systemic autopoeitic properties that are more than the sum of its parts, we develop an approach of modeling the smartness of the city. To understand how the city may behave as a sustainable ecosystem, we need a framework to design the interactions of the city subsystems. *First* we define a smart city as an ecosystem that is more than the sum of its parts, where sustainability is maintained through the interactions of urban functions. *Second*, we present a methodology to sustain the development over time of this ecosystem: Urban Lifecycle Management. *Third,* we define the tasks to be carried out by an integrator of the functions that constitute the smart city, we assume public administration has to play this role. *Fourth*, we present what should be a smart government for the smart city and the new capabilities to be developed.

1 Introduction

At date, there is no standardized definition of what a smart city is, in spite many apply to propose a definition that fit with their offer, subsuming the whole of the city in one of its functions (smart grid, smart mobility…). *First* we define a smart city as an ecosystem that is more than the sum of its parts, where sustainability is maintained through the interactions of urban functions. *Second*, we present a methodology to sustain the development over time of this ecosystem: Urban Lifecycle Management. *Third,* we define the tasks to be carried out by an integrator of the functions that constitute the smart city, we assume public administration has to play this role.

Fourth, we present what should be a smart government for the smart city and the new capabilities to be developed.

This paper is based on case studies carried out within the cluster Advancity (France) for the urban ecosystem issue, and other case studies on the intention to design new business models based on the concept of extended enterprise and extended administration. It relies on the state of the art in complex system architecture as developed in information system and system engineering in complex products such as aircrafts, to envisage how these competencies may be adapted to public services in their collaborative work with private firms.

2 What is an urban ecosystem?

A smart city is more than the sum of "smarties" (smart grids, smart buildings, smart computing…) but there is not, at the present time, a precise and operational definition of what a smart city is (Lizaroiu & Roscia, 2012). Several pretenders exist on what a smart city could be (Songdo in Korea, Masdar in Abu Dhabi,…) but they are not cities to live in, they are demonstrators, propelled by big companies (e.g. Cisco in Songdo) who apply a particular technology to the conception of a city.

In the literature, the smart city is recently defined as an ecosystem, that is to say a system where the whole is more than the sum of the parts and has autopoeitic properties (Neirotti et a., 2013).

For the systems architect this approach implies:

> · Defining a perimeter that comprehends all the components that have a critical impact on city life: the city needs to be fed, imports products that may have been manufactured on a basis that does not fit with sustainable development requirements (pollution, children work or underpaid workers, carbon emissions…). These costs and environmental impact must be charged to the city balance.
>
> · Considering the system as a living system where the behavior of inhabitants determines the sustainability of the ecosystemic properties of the city. The underlying assumptions are material systems in addition to immaterial ones – as history, culture, anthropology and social capital – play their role. A recent trend in the literature on development economics, which is contrary to the fad of mainstream economics that consider all territories alike, put the emphasis on the "smart territory" as an unstructured cluster of tradition, culture, and informal institutions able to shape an innovative milieu (Aydalot, 1986).

If the city is an ecosystem, according to the laws of general system theory (Ashby, 1962) it may be represented as shown in figure 1:

> A) It has a finality made of strategic vision borne by stakeholders (public and economic actors), people living in the city and sustaining this finality through theirs activities, and preserves its identity by interactions with its environment.

B) This system may be broken down in tree structures of subsystems: the functions. These functions belong to hard and soft domains. Hard domains include energy, water, waste, transport, environment, buildings, and healthcare infrastructures. Soft domains include education, welfare, social capital, public administration, work, civic activity and economy. What makes the city intelligent is the richness of connections between branches. We speak of a tree structure here in the sense of Herbert Simon's architecture of complex systems (1969) where the designer will connect the subsystems to make the system emerge according to the aim it pursues. In his seminal paper "a city is not a tree" (1965) Christopher Alexander, an architect initially trained as mathematician and Professor at Berkeley, criticized the conception of the urban planning movement in America, considering it as a "fight against complexity", with no connections between branches. Modern cities conceived for cars, compared to ancient cities, offer a very poor web of connections. Alexander formalized his idea of the city conceived as a rich overlapping of building blocks in his 1977 book *A pattern language*. This insight of considering the whole as a combination of modular and reusable building blocks lingered on the margins of architecture but has had an enormous influence in the development of object oriented architecture in software design.

C) These functions are operated using tools and artifacts of which end-users are people, specialized workers and ordinary citizens. The critical point is that people must not fit the tools but, on the contrary, tools and artifacts will fit to people only if the right societal and institutional conditions are met.

Modeling the ecosystem implies answering three questions (Krob, 2009):

- The first question is WHY the city: what is the *raison d'être* and what are the goals of the city regarding WHO are the stakeholders and which activities will support it? Beginning with this question may avoid the drift towards a techno pushed approach relying on technological determinism, one may find in Songdo or Masdar.

- The question why is then deployed in questions WHAT: What are the function the smart city must perform to reach these goals? These functions are designed in processes grouped in subsystems aligned with the goal of the main system.

- The third set of questions concern HOW these functions will be processed by technical organs operated by the people who are the city executives and employees, and the city dwellers as end users.

The issue is not to define an ideal type of smart city since all the "fitting conditions" that make the city smart will be different according to the context, but to define modeling rules to conceive and sustain the ecosystem.

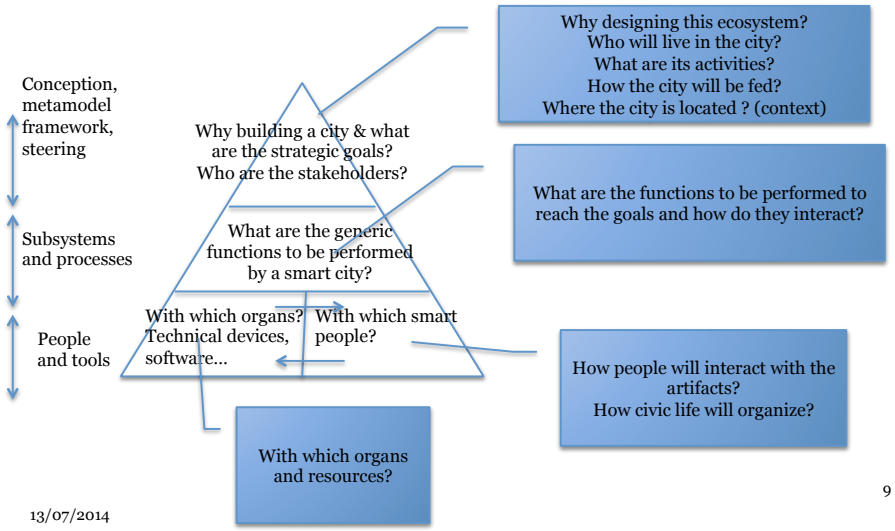


Figure 1.0: architecting the ecosystem

3 The global framework: Urban Lifecycle Management©

Since the advent of the "death of distance" with the revolution of transportation by the middle of the XIX° century, the appearance of networks of infrastructure technologies and the spread of the telegraph that transformed the government of the city, critical obstacles to the growth of cities were removed. Today digital technologies amplify this move, providing new tools such as smart phones that became a digital Swiss knife that allows inhabitants to be active actors in the city life, communicating and coordinating with each other, using and feeding databases. Doing this, digital technologies may produce the best and the worst. The point is each city contains the DNA of its own destruction. Smart cities digital infrastructure amplifies the possibilities of manifestation of discontent, worsening the gap between have and have-nots. Smart cities incur the risk to become the digital analogue of the Panopticon Jeremy Bentham's prison design (Townsend, 2013).

We assume that the rules of complex system modeling and system architecture may apply to the city as well as they apply to products through PLM (Product Lifecycle Management) in that case according to a framework we call Urban Lifecycle Management© (ULM). The difference is a city never dies and must permanently renew its economic and social fabric as well as its infrastructure. An unsmart city will continuously expand according to the laws identified by G. West and L. Bettencourt (2007) that reveal an increasing return in infrastructure investment that allow the city to sprawl indefinitely. The complexity will grow out of control, resulting in a city

being the sum of heterogeneous boroughs with strong social and economic heterogeneity and spatial dystrophy.

We define ULM first and foremost as a tool to design an ecosystem which will be coherent with the political, social and economic goal people assign to the city according to the principle of sustainable development: stability, waste recycling, low energy consumption, and controlled scalability, but in a way that allows to foresee its evolution and to monitor the transition in different ages of the city.

ULM has to counterweight the appeal of technological determinism: in the past, technologies have always dwarfed their intended design and produced a lot of unintended results. ULM has to monitor the life of the smart city alongside its evolution, as represented in figure 2.0

- A city can't be thought out of its historical and cultural context that is represented by the territory of which the city is the expression. The smart city embarks a strategic vision that is based on a strategic analysis of the context and material and immaterial assets of the territory (GREMI, 1986). The smartness of a city profoundly relies on what has been coined as "social intelligence" by prof. Stevan Dedijer in the years 1970s as the capability to build consensus where each social actor relies on others to create new knowledge. Intelligence doesn't operate in a vacuum but is socially and culturally rooted (1984).

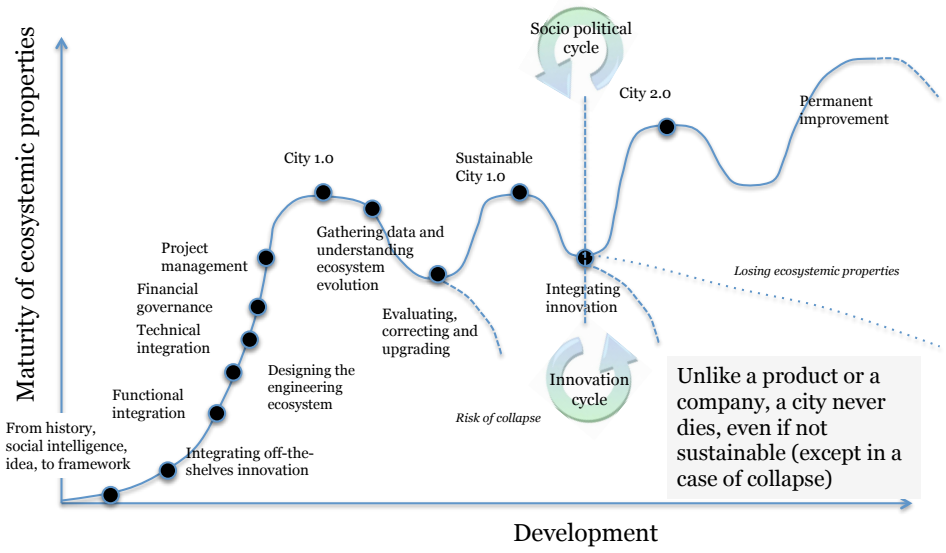


Figure 3.0: Urban Lifecycle Management©

- To be livable, the city may not be a prototype city: the system architect must focus on the task of integration that needs to be reliable to proceed from off-the-shelf components that already have an industrial life and may be

considered stable and reliable, in the same way the classical architect does not invent the brick in the same time as he designs the house. This will imply coordination between innovation cycles as we will see further.

- The process carried out on the principles represented in figure 1 leads to a first release of the city 1.0 in case of a new city. Just as well in a new or old city, we need to understand how the city lives and the unavoidable discrepancies between intended design and real result, an observatory must be implemented that will collect data produced by the city. Corrections are made according to classical principles of quality process management.

- Alongside the lifecycle, exogenous innovation will occur that will need to be endogenized by the model. For example, Songdo in his initial design relied on RFID devices to track city dwellers. Today, smart phones have become the Swiss knife of the city dwellers, rendering the use of RFID devices obsolete. Innovation is ubiquitous in all subsystems of the city. Innovation in smart cars interacts with the architecture of transportation (hard subsystem) as well as in human behavior (soft subsystem). Coordination will be needed through common frameworks such as projects management office extended to the global smart city's complexity.

- Innovation challenges the equilibrium of the smart city. Not all innovations are compulsorily good for the city: Civic and political life have to evaluate the consequences of an innovation and to frame it so that it fits with the common good and the sustainability of the city.

- All along its lifecycle, the city may lose its smartness with two undesirable consequences: the city may continue to sprawl on a non-sustainable basis leading to today clogged cities. In case of a disruption in its core activity, the city may collapsed as it happened in the past when things become too complex to be monitored, as studied for past civilizations by archeologist Joseph Tainter (1990). Reducing the size of the city is then the only solution to reduce the complexity. A similar thing appears today in Detroit, a city that has lost its goals and population, leading to the decision of reducing the size of the city as the only means of avoiding bankruptcy. A similar pattern exists with the Russian monocities.

4 The rationale for extended public administration in the process of integration in ULM

No two cities are alike however smart they are, but the principles of system architecture ULM are based on the assumption that common rules of modeling may be defined. One of the key rules is to understand the interactions between economic development and human capital: economic development is critical to draw financial resources for investment in new transportations, infrastructures and education. Cities with a greater economic development appear more attractive to people who wish to increase their standard of life and who are more fitted to increase the smart cities

human capital. The more a smart city has a high level of human capital, the more she has end users able to develop, test and use new tools that improve the quality of urban life (Neirotti & a. 2014). It is all the more true in the digital era were the end-user is not only a consumer but also a prod-user – according to the definition by sociologist Axel Burns – who is involved in a continuing process of producing never finished artifacts. On the other hand, the city has to take care to not create a digital divide.

The modeling rules consist of three main principles:

1. **Strategic analysis:** As represented in figure 1.0 the first task is to define the issues with the stakeholders. The functions needed to reach these issues are then defined, and deployed in organs and specific competencies and resources, as represented in figure 3.0

2. **Inventorying the building blocks**: There is no absolute definition of what is a smart city is and in spite we may define general rule of modeling, the definition of the smartness of a city will always be specific to the context, e.g. geographical and climate constraints (a city exposed to tropical floods or earthquake will embark functions that a city in a temperate country won't need), economic activity (specialization, search for synergies, position on the commercial routes and worldwide supply chains). The selection of these functions is essential to build a resilient city, e. g. with the climate change new phenomenon occur such as flood, marine submersion, extreme frost the city was not prepared for.

Nevertheless, common functions will exist in every city and their organization may proceed from off-the-shelf patterns.

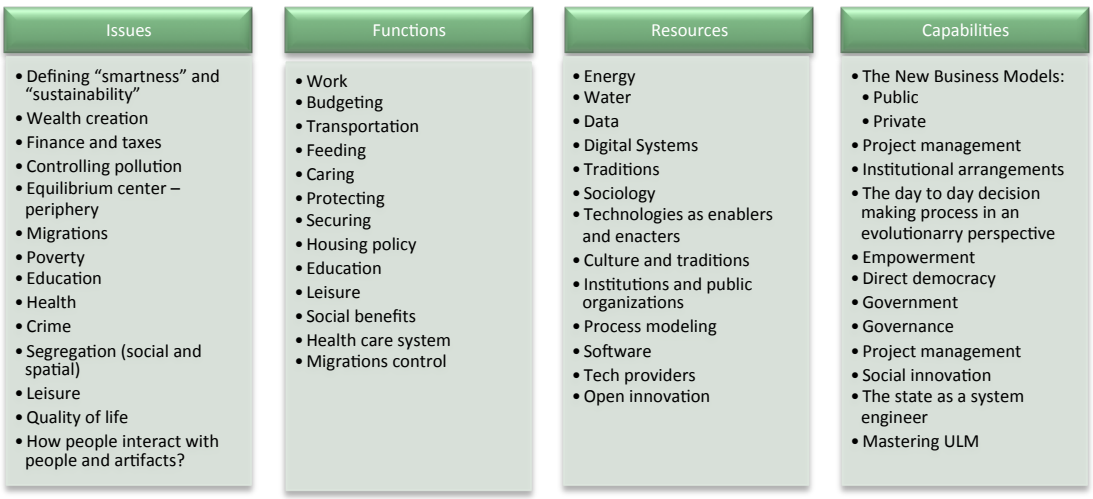
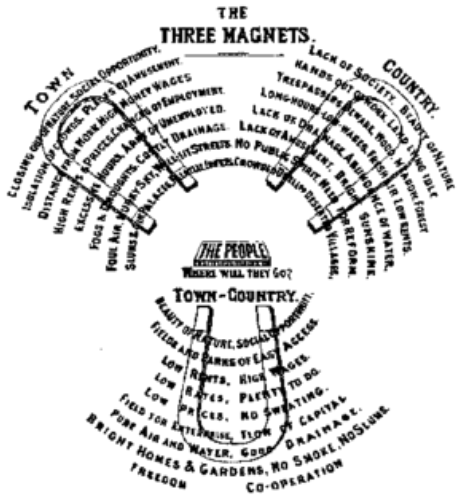


Figure 3.0: The building blocks

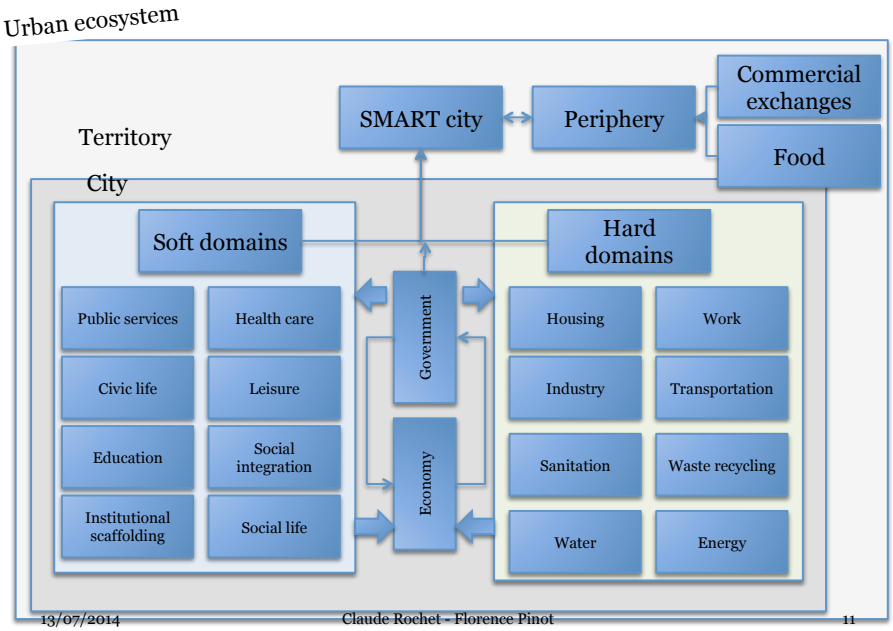
3. **Integrating the ecosystem**: In complex systems dynamics, the behavior of a system as a whole is an *emergence*, that is to say that the property of the system can't be attributed to one function in particular but is the result of interactions between these functions. The "good life" is the basic question of political philosophy since Aristotle. It is an ethical issue that will result from political and strategic debates among the stakeholders. Jane Jacobs (1995) has criticized the utilitarian approach that prevailed in America in the city planning movement. The ancestor of the urban planning movement, Ebenezer Howard, thought of the smart city as an ideal city conceived from scratch as a mix of country and city. His insight was to conceive the city as an interaction between a city with jobs and opportunity but with pollution, and the countryside with fresh air and cheap land but with fewer opportunities, each one acting as magnets attracting and repelling people. He invented a third magnet, the Garden city, which combined the most attractive elements of both city and countryside (Howard, 1902). Garden city was the Songdo of its day (Townsend 2013) that galvanized architects, engineers and social planners in search of a rational and comprehensive approach of building city. Howard's approach was excoriated by Jane Jacobs in his *Death and Life of Great American Cities* (1961) for not giving room to real life: "He conceived of good planning as a series of static acts; in each case the plan must anticipate all the needed… He was uninterested in the aspects of the city that could not be abstracted to serve his utopia". In fact, the city garden dream, not relying on a global systemic architecture, has degenerated in the banal reality of suburban sprawl.

The same risk exists today with digital technologies, which could revive the ideal city dream, under the impulse of the big players such as Cisco, IBM, Siemens, GE who have interest in a top-down and deterministic approach that reduce smart cities to the adoption of their "intelligent" technology. To avoid this bias system architecture must be on the top of the agenda of extended public administration. This activity may be summed up in four points:

a) Soft and hard subsystems: Today's prototypes of would be smart cities are techno pushed and put emphasis on the possibilities of technology to make the city smart but mainly forget the inhabitants. City dwellers have the main role to play since it is their behavior and their use (and more and more the production) of information and technology that make the day to day decisions that render the ecosystem smart or no. Figure 4.0 represent both parts of the ecosystem the *soft* or human subsystem and the *hard* one,



the group of technical subsystems. Integration of these subsystems obeys different laws: human subsystems are dissipative ones, difficult to model, not obeying physical laws with important entropy. Reducing their uncertainty relies on the sociology of uses, social consensus based on accepted formal and informal institutions, and a close association of inhabitants to the design of the system, which is a common feature of complex system design. Physical subsystems are conservative ones that can be modeled through the laws of physics with a possibility to reduce entropy, but keeping in mind that the decider in last resort is the city dweller who will use it.



• *Figure 4: The smart city as an emergence*

b. **Outside/inside**: The urban ecosystem is not reducible to the city itself, with perhaps the exceptions of city-states as Singapore where the limits of the city are given by nature. A city must be fed and have exchanges with a close periphery which produces goods (services, agriculture, food…) in interaction with the center. The design of a system relies on the definition of the border of the system. According to the laws of complex system modeling (Ashby law) the inner complexity of a system must be appropriate to the complexity of its environment. So, the urban ecosystem will have to define three perimeters: the first is the city itself inside which the synergies and interactions are the stronger and have the most "eco" properties. The second is the periphery: one may refer here to the model defined by Thünen at the beginning of the XIX° century representing the city with a succession of concentric rings going from the highest increasing return activities at the center city to decreasing return activities at the periphery (Schwarz, 2010). The first represents the exchanges

between the ecosystem and the rest of the country. This represents logistic costs that may have a negative impact on pollution and carbon emission that may be reincorporated in the balance of the city to measure it smartness. The third is the external environment with witch the city exchanges, that is, in a age of a globalized world, the rest of the world: the larger this perimeter, the more the system is subject to external factors of instability and the less the ecosystem is coherent as a Thünen zone[1].

c. Combining top down and bottom-up integration: Each industry has today its model for the integration of its activities. Smart grids, water suppliers, transport operators, IT providers … have model for systemic integration of their subsystem and to evaluate its impact on the global functioning of the city. On the other hand, we know that the urban ecosystem being more than the sum of the subsystems we need another approach that starts from the top, that is from the strategic goals of the city deployed in functions as represented in figure 1.0. Where will be the meeting point of these two approaches? Proceeding bottom-up will raise problems of system interoperability, data syntax and semantics, while the top-down approach is more relevant to define strategic issues but will have to integrate all the existing businesses and functions. A possibility is that storing data in common data warehouses and completing it with the exploitation of big data will provide common references. In any case, the answer will proceed from applied research projects in building cities.

d. Defining new business models and competencies: Conceiving ecosystems needs the enterprises to cooperate to share a common strategic view so as to form a conception ecosystem based on the principle of "coopetition" (cooperation *and* competition). Each enterprise must define its performance indicators according to the performance of the whole and not only to that of its parts. The same concern is for public management: with the silo organization of public administration, no one is in charge of a global view of the city. This calls for new business models of enterprises extended not only to the partners of one enterprise but to the global value chain of the ecosystem. The same applies to public administration in its very organization to develop the competencies needed to deal with complex system design as well as its strategic thinking. The French public administration still consider its industrial strategy in terms of "*filières*" (channels) that are the vertical integration of similar activities (such as aerospace, automotive…), as it was relevant in the paradigm of mass production, while the locus of disruptive innovation is in the overlaps of different industries.

The French government was baffled when GE announced his intention to take over Alsthom. Would the French administration have understood the strategic issues at stake with smart cities as ecosystem and not only with the hard subsystems (water, sanitation, transportation…) where France has traditionally

[1] We may give as an example the city of Quimper at the heart of the granitic massif of Brittany (France) who choses to import its granite from China.

strong positions, she would have valued differently the smart grid activity of Alsthom and its interest for competitors who aim to preempt the smart city market which value is estimated, for the sole so-called smart infrastructures, at 100 billion USD for the coming decade (Townsend 2013)[2].

Another strategic issue is the battle for norms: a smart city is not, at the present time, defined with norms, metrics and metrology. Defining the norms (in terms of ISO standards) will allow lock-in the conception of smart cities by shaping all the tenders.

5 Smart government, the keystone of smart cities

Far as back as 1613, the Napolitano Antonio Serra, in a memoir presented at the vice-king of Naples, analyzed the city as the place where activities with the biggest increasing returns take place, with a strong correlation between economics and politics (Reinert S., 2011). The frescoes of the Siena town hall by Ambroggio Lorenzetti depict "the good government" as a dynamic equilibrium between intense economic activities and an active political life that gives the people of citizens the power to rule the city according to the principles of the common good. Contemporary evolutionary economics correlates the evolution of institutions with that of economic activity (Reinert E. 2012).

The growing complexity of cities and the predominance of top-down urban planning have made us forgetful of these lessons from the past. In their analysis of present smart cities initiative, Neirotti & a. (2013) notice that there is no practice that encompasses all the domains, hard and soft, of the cities. On the contrary, the most covered domains are hard ones: transportation and mobility, natural resources and energy. Government is the domain in which the cities report the lowest number of initiatives. More, there is an inverse correlation between investment in hard and soft domains, and smart government is still the poor relative in smart cities initiatives and cities that have invested in hard domains are not necessarily more livable cities. In fact, two models emerge form Neirotti & a. survey: one focused on technology (with a strong impetus of technology vendors) and one focused on soft aspects, the hard model being dominant. The problem is there are no vendors for soft domains apart the citizens themselves whereas systemic integration relies on soft domains, mainly taking in account the context and valuing social capital.

Smart cities conceived as ecosystems should provide policy makers with some practical guidelines to integrate soft and hard domains. Three areas for smart government appear:

[2] The total market of smart cities is estimated as much as $350 trillion needed to build, maintain, and operate the world's cities over the next forty years (*WWF report "Reinventing the Cities", 2012*)

Economic development: In the past, smart cities have been built without central planning (except in the case of Roman cities which reflected the imperial objective of the Roman Empire) but with a clear, although not explicitly formulated, founding purpose: defense, commerce, religion, power, geography… The pattern of the city emerged out of the interactions of key stakeholders: The lord, the barons, the merchants, the shopkeepers, the craftsmen, the bankers and the people. The design of ancient cities made them intelligent since they were ecosystem that sustained and reinvented themselves along time… till the point their capacity to self-reinvent came to an end when the core of their strategic activity reached a tipping point (e.g Italian cities after the Renaissance, Russian monocities from the USSR era, Detroit today). The design of these cities obeyed to the real interactions underlying economic life (roads, markets, fairs, harbors, work, industry…) and civic activities (agora, city hall, structure of power). Their global ecosystem may be referred to as the ideal type conceptualized by J.H Thünen at the beginning of the XIX° century, that is to say a center where the core of the city is with the strongest interactions and the returns are the highest, surrounded by concentric zones going of decreasing returns activities (Schwarz, 2010).

The task of government is to search for the activities that produce the highest increasing returns, no thanks to high technology but to synergies between activities (Reinert, 2012), that will constitute the center of the Thünen zones. The Russian monocities built on a unique industry (coal, oil, cars, aerospace….) linger as long as this industry has a leading role but have very poor capabilities to reinvent itself due to the lack of synergies between different economic activities.

A vibrant political life: With cities emerged political philosophy. The most perspicacious analyst of what makes a city great was undoubtedly Machiavelli who put emphasis on the necessity of the common good : "*it is the common good and not private gain that makes cities great* » he wrote in his Discourse on Livy. Machiavelli conceived the common good in the Thomas Aquinas' tradition as a whole superior to the sum of its parts. Its systemic equilibrium is permanently challenged by the corruptive forces of *fortuna* that must be offset by the *virtù* of the Prince and the dynamism of the *vivere politico* (Rochet, 2010). Emphasis has been put on the topicality of Machiavelli to understand the systemic character of public management (Rochet, 2009). The vitality of the system is sustained with permanent interactions within thanks to a vibrant political life that provide a space for controversies. Machiavelli praised the Roman republic for his institution of the tribunate that managed the confrontation between the many of the citizens and the few of the ruling class that allowed the Republic to upgrade his institutions according the principles of the common weal advocated by Cicero. The conservative French politician and historian François Guizot attributed the success of the European civilization to the permanence of the classes struggle as a means to build political compromises as a guarantee of sustainability, under the conditions that no class wins. In contemporary complex societies, Elinor and Vincent Oström have developed the concept of polycentric governance that is organizing governance on one hand on a vertical axis from upper to lower levels of complexities, and on the other hand on an horizontal axis which consists of overlappings between organizations (Östrom, 2010). Elinor and Vincent Ostrom have criticized the excess of rationality that defines strict boundaries

within missions and attributions of public organizations, since the reality doesn't know these boundaries and the adaptive character of public systems may be found in their overlaps.

Supporting open innovation: In the contemporary smart cities, information technologies give more power than before to citizens to use and produce information, and also applications. The experience of cities opening their database to the public to trigger the development of apps has proved the payoff of bottom-up approaches: in Washington DC, a contest "apps for democracy" challenged the local developers to create software exploiting public resources. For a cost of 50 000 US$ the pay-off was blazingly fast with forty seven apps developed in thirty days, representing an estimated 2 million worth of services, about 4000% return on the city investment (Townsend 2013).

But one should not conclude that bottom-up approaches are the killing solution: theses apps are V 1.0 developed by techies on the basis of a fascination for technologies while the city needs V 7.0 tested and reliable and based on the real needs and problem solving of citizens as end-users not familiar with technology. We rediscover here one of the law of innovation emphasized by Von Hippel (1986): the key role of lead users in the innovation process which is furthermore not a specific aspect of innovation in the digital era but a permanent, although forgotten, feature of the innovation process in the industrial era as reminds us François Caron, a leading academic in history of innovation (Caron, 2012).

In the same manner national innovation systems exist (Freeman, 1995) and provide a framework that gives incentives to cooperation between industry, research and investors to steer their activities toward risk taking innovations, extended public administration could structure an urban innovation system that would structure the innovation process in a way that would guarantee that innovation, research and development of so-called smart apps are focused on the real needs of the city dwellers.

6 Conclusion: Extended administration as art of systemic integration

In the absence of a definition of how intelligent may be cities to be sustainable, today's initiatives are techno-pushed since tangible goods of the hard domains of smart cities drive the market. Digital economy seems to be the keystone of smart cities, but we have shown that the keystone in last resort is the end-users of technologies: the citizens. This requires a combination between soft and hard domains that can be achieved through complex systems architecture (Godfrey, 2012), a new discipline, methodology and competency in public management that we coin as urban lifecycle management©.

Although according to system theory self-regulating systems exist – but once their genetic codes have been written - as they exist in nature and in small-scale human system such as those studied by Elinor Östrom for the management of the commons (Östrom, 1991), large complex systems such as smart cities need to be framed by a

central architect before reaching its resilient and sustainable stage. The newborn concept of extended administration finds here its application in its intention to encompass and to design the global value chain of public administration and its interaction with – and between - all the stakeholders. This implies a sea change in the competencies and business model of public administration. This new field would be carried out through research in action projects building cities as ecosystem tending toward resilience where humans are first to decide for the ends.

References:

- Alexander, C. (1977) "A pattern language, town, buildings, constructions", with Sarah Ishikawa et Murray Silverstein, Oxford University Press
- Ashby W.R. (1962): "Principles of the Self-organizing System", in:Principles of Self-Organization, von Foerster H. & Zopf G.(eds.),(Pergamon, Oxford), Cambridge, MA, 193-229.
- Aydalot Ph. Ed., "Milieux Innovateurs en Europe", GREMI, Paris, 1986.
- Caron, François (2012) « La dynamique de l'innovation », Albin Michel, Paris
- Dedijer, Stephan, 1984 « Au-delà de l'informatique, l'intelligence sociale », Stock, Paris
- Freeman, C. (1995a), 'The national system of innovation in historical perspective', in *Cambridge Journal of Economics*, vol. 19, no. 1.
- Godfrey, Patrick, « Architecting Complex Systems in New Domains and Problems : Making Sense of Complexity and Managing the Unintended Consequences » in *Complex System and Design Management, Proceedings, 2012*.
- Hardin, G., The Tragedy of the Commons, *Science, New Series, vol. 162, n°3859 (dec. 13, 1968), pp. 1243–1248*.
- Howard, E, 1902, "Garden Cities of To-morrow" (2nd ed.), London: S. Sonnenschein & Co
- Jacobs, Janes, 1985 « Cities and the Wealth of Nations », Random House, New-York.
- Krob Daniel, Eléments d'architecture des systèmes complexes, [in "Gestion de la complexité et de l'information dans les grands systèmes critiques", A. Appriou, Ed.], 179-207, CNRS Editions, 2009.
- Lizaroiu G.C, Roscia M. 2012 « Definition methodology for the smart cities model »
- Neirotti P., De Marco A, Corinna Cagliano A, Mangano G, Scorrano F, "Current trends in Smart City initiatives: Some stylised facts" *Cities, Volume 38, June 2014, Pages 25-36*

- Ostrom, Elinor, 1991 « Governing the Commons ; The Evolution of Institutions for Collective Action » Cambridge University Press, NY.
- Ostrom, Elinor, 2010 "Beyond Markets and States: Polycentric Governance of Complex Economic Systems" *American Economic Review, 1-33.*
- Reinert, Sophus A., ed. Antonia Serra, "A Short Treatise on the Wealth and Poverty of Nations (1613)". Anthem Press, London 2011.
- Rochet, Claude, 2008 Le bien commun comme main invisible : le legs de Machiavel à la gestion publique, *Revue Internationale des Sciences Administratives, 2008/3 (Vol. 74)*
- Rochet, Claude, 2011 "Qu'est-ce qu'une bonne décision publique "? Editions universitaires européennes
- Schwartz, Herman, 2010, « States vs Markets : The Emergence of a Global Economy », Palgrave 3rd ed.
- Simon, H. A., 1969, « The Sciences of the Artificial », MIT Press, 3rd ed. 1996.
- Tainter, J. 1990 "The collapse of Complex Societies" Cambridge University Press.
- Von Hippel, E., 1986, "Lead Users: A Source of Novel Product Concepts", *Management Science, 32(7): 791–806,*
- West, Geofrey, Luís M. A. Bettencourt, José Lobo, Dirk Helbing, Christian Kühnert « Growth, innovation, scaling, and the pace of life in cities » Indiana University, 2007.

Active Experimentation and Computational Reflection for Design and Testing of Cyber-Physical Systems

Kirstie L. Bellman[1], Phyllis R. Nelson[2], and Christopher Landauer[1]

[1] Topcy House Consulting, Thousand Oaks, CA USA
[2] California State Polytechnic University Pomona, Pomona, CA USA

Abstract. Cyber-physical systems are being deployed in a wide variety of applications, creating a highly-capable infrastructure of networked "smart" systems that utilize coordinated computational and physical resources to perform complicated tasks either autonomously or in cooperation with humans. The design and testing of these systems using current methods, while time-consuming and costly, is not necessarily sufficient to guarantee appropriate and trustworthy behavior, especially under unanticipated operational conditions. Biological systems offer possible examples of strategies for autonomous self-improvement, of which we explore one: active experimentation. The combined use of active experimentation driven by internal processes in the system itself and computational reflection (examining and modifying behavior and structure during operation) is proposed as an approach for developing trustworthy and adaptable complex systems. Examples are provided of implementation of these approaches in our CARS testbed. The potential for applying these approaches to improve the performance and trustworthyness of mission-critical systems of systems is explored.

1 Introduction

Complex systems of systems (SoS), especially those that include cyber-physical systems (CPS), are now being deployed in critical infrastructure applications such as the electrical grid, health care, manufacturing, transportation, commerce, law enforcement and defense. We bet our lives, or at least our livelihoods, that these systems will function as anticipated. Yet, as they become increasingly complex and interconnected (networked), developing the systems engineering methods to ensure that these SoS will be trustworthy has become its own technical challenge.

For example, space systems (which term includes not only the satellites, but also the ground control and dissemination systems and the launch systems that put them up there) are simply the most complex engineered systems that humans build that work (and they do work almost always and often far beyond their projected design life). They typically involve hundreds of organizations, thousands of people, tens of thousands of components, millions of pages of documentation,

and they are expected to last sometimes for decades. (The development process does usually last for decades even when the satellites are not expected to). It has been clear for some time that these systems exceed our ability to understand them, and that they only work by dint of what we have heard called "heroic engineering", but even that approach is now regularly exceeded by current and planned systems.

Successfully planning and implementing the integration of such complex constructs would, in principle, require detailed knowledge of hundreds of thousands (or more) of components, how they are connected into subsystems, and all of the possible interactions between components, subsystems, and the environment. From a practical perspective, it is exactly the lack of this detailed knowledge that leads us to characterize a system as complex. [22] Various approaches based on formal compositional methods [10, 11, 21, 24, 25] or brokering of mutual requirements (service-oriented architectures) [2, 23] have had some success, but these approaches do not adequately address the central problem: precise descriptions of all of the components to be integrated, and especially all of their possible interactions, are not fully known, and therefore not available for use in the design and integration processes. Existing approaches do not enable the discovery of the new knowledge that is needed to guarantee appropriate functioning of the integrated SoS.

Systems of systems are built from systems that themselves have been developed and tested, often for a different application. The integration challenge, then, concerns most importantly the necessity of reconciling the multiple and sometimes conflicting operation and control strategies of these systems with respect to a new SoS purpose or goal. [6] Conflicts in which a component system continues to operate in accordance with its own best interests given the previous application may no longer allow the full SoS to operate as needed, but these conflicts are difficult to discover without testing the full operating SoS. Therefore, the testing required for verification and validation of the operation of the full SoS is potentially damaging to the SoS itself, and also risks interruption of the services it supplies. This paper proposes a strategy by which active experimentation coupled with computational reflection can refine or even discover the knowledge needed to ensure appropriate functioning of the overall SoS.

2 Systems Engineering Challenges

Complex systems of systems challenge established systems engineering practices in several ways.

- Managing the complexity is a fundamental technical challenge in itself, independent of the particular system or application.
- Updates and upgrades mean that the SoS evolves during its operational life.
- The capability and value of a system / component / device leads us to repurpose it for applications that were never envisioned by its original designers rather than developing a completely new device.

- Instances of the system are often unique, although there may be other, similar instances (i.e. Amtrak's reservation system, a segment of the electrical power grid, a space system including all ground and launch resources).
- Ubiquitous wired- and wireless communications networks mean that the boundaries of the SoS and its possible states are probably not completely definable.
- Self-x capabilities mean that the system is never fully designed.

Component, subsystem, and system design and test currently utilize a variety of models at differing levels of detail, together with a set of "goodness" measures linked to *a priori* requirements, as inputs to computational processes (often optimization) that evaluate candidate strategies. However, as complexity increases, "emergent" behaviors become increasingly likely. Such self-organized, coherent actions that were not planned or anticipated by the designers often occur through interactions that are not present in the models of components, processes and interactions used in design, integration and test. CPS SoS design and testing is further complicated by the re-purposing of legacy hardware and software which may not have been designed in accordance with current procedures, standards and interfaces, thus requiring specialized adaptations. New approaches are needed for design, verification and validation of complex cyber-physical SoS to better ensure their trustworthiness. The most desirable of these approaches will also address the spiraling cost of implementing and testing these complex SoS.

3 Biologically Inspired Control Strategies

Biological systems provide a rich source of inspiration for engineering complex SoS both because, in spite of their obvious complexity, they achieve remarkable robustness, and also because of the extreme degree of interconnection of their various components and subsystems. [5, 9] A central lesson that we have taken from biology is that both robustness and controllability can result when each component or process interacts strongly with many other components and processes in a monitored and regulated system. (A recent example comes from work on the immune response of the mammalian gut microbiome, [1, 20] but there are many others.)

Control in biological systems occurs through the combined operation of many processes and actions, with desired behaviors being achieved by small changes in relative strengths. A web of overlapping monitoring and regulatory processes that maintain appropriate conditions at all levels of complexity is critical to the success of this paradigm. For example, the actions and processes used to achieve the top-level goal of walking over rough terrain are achieved by many instances of humans in spite of significant differences in their structure, strength and ability. That is, biological systems rely not on uniformity of structure, but on the ability to adjust similar structures and generic patterns of actions based on a high degree of monitoring of local conditions in order to accomplish a behavior that is adequate for the current context and goal.

Controlling a SoS with a complex web of balanced interactions is strikingly different from the traditional block-diagram approach to engineering design that focuses on building a few strong and well-understood interactions between components while striving to nullify all other interactions. We suggest that the assumption that small interactions can be neglected, together with implementation of this assumption throughout the modeling process, is one important reason that emergent behaviors are often not predicted by simulations. In contrast, the biological style does not deprecate interactions, but instead achieves a "balance of forces" form of control based on extensive overlapping webs of monitoring and regulation at all levels of the hierarchy of complexity. We propose that implementing this style in strategic portions of engineered systems could mitigate the challenges posed by unmodeled interactions.

The biological design approach leads to a "permissive" style in which, while actions, states, conditions, and processes may vary from one instance to another, overall performance goals are achieved by adjustments in their relative intensities. This permissive style is in clear contrast to the restrictive control approach of traditional engineered systems, in which adjustments of a few inputs achieve all of the desired actions or processes through clearly defined pathways. However, there is promising similarity between the dense web of interactions in biological systems and the challenge of managing the many unknown or unmodeled interactions in a complex SoS, again suggesting that a more biological approach to design, operation, and integration may be useful provided that the appropriate information about actual interactions can be discovered.

The admirable robustness of biological systems is due in part to their ability to learn to accomplish the same goal using a variety of strategies, although not necessarily equally efficiently. For example, if you break your right arm you are still able to accomplish most of the tasks of daily life by substituting your left arm or accommodating to the reduced motion allowed by a cast. This broad ability to find a way to accomplish a goal in spite of changes in capability or configuration is exactly the type of robustness and reliability that we would like to have in engineered SoS, and to understand and utilize during design and integration. New ways of acting can take place through the recruitment of existing structures and processes in new combinations to address a new context, purpose or goal. [3] Thus, a large space of possible responses can result from small departures from previous conditions. Since this style of operation differs significantly from the usual engineering approach with narrowly defined and targeted control pathways, new tools and methods are required in order to exploit it effectively.

4 Active Experimentation and Computational Reflection

The success of the biological control-through-balance style rests on experience with the available processes, structures and patterns, as well as of their limits of capability and their applicability to situations similar to the present one, either through evolutionary selection or from the experience of a specific individual.

This knowledge is not necessarily innate in a biological system, just as there is important knowledge lacking in models of SoS.

Biological systems use excess resources to actively experiment. By doing so, they discover and refine models of their capabilities, limitations, and possible interactions with their surroundings that include consideration of both internal state (hungry, cold, tired, etc.) and external conditions. Significantly, such experimentation also enables the grouping of collections of useful resources, processes and capabilities into generic pre-patterned templates with simplified control mechanisms. Such templates can be easily shaped to fit a specific current context. [7, 8]

The biological analogy suggests that, if it were possible for a complex SoS or some of its components and subsystems to engage in active experimentation, the existence of conflicts between the existing operation and control strategies of a repurposed subsystem and the overall SoS purpose or goal could be identified and modeled before such a conflict gives rise to failure or to disruption of the service provided by the overall SoS. In addition, efficient strategies for accomplishing common purposes and goals could be discovered and collected into templates for accomplishing similar operations. Such templates could then be reviewed for correctness either by the system itself using further active experimentation, or supplied to designers, integrators and operators for evaluation.

Our existing systems have not been built with the capabilities required in order to engage in active experimentation. Thus, an important research challenge is to implement such processes while preventing the resulting experiments from damaging some part of the SoS or compromising the service it provides. Implementation is particularly challenging when, as with space systems or the electrical grid, there is only one operating instance of the entire SoS.

In following sections we discuss several possible approaches for introducing active experimentation into engineered complex cyber-physical SoS. However, first we list the additional capabilities required of such implementations. They are:

- instrumentation at all levels of the hierarchy of complexity to measure what happens.
- models that relate what is measured to properties or symbols that are local but have meaning that can be communicated to other parts of the SoS.
- models that relate what is measured locally to higher-level purposes, goals and constraints.
- the capability to retain the information produced by these measurements and models.
- a hypothesis-generating engine that can propose possible actions (experiments).
- a predictive capability to project and analyze the potential consequences of a proposed future action.
- the ability to engage in a proposed action.

Taken together, these resources and capabilities would create an engineered SoS able to reason about itself (its resources, capabilities, and limitations) in the

context of its current environment, purposes and goals, and also to both propose and implement a course of action based on that reasoning rather than on pre-programmed control strategies. [13, 15]

These capabilities required for achieving active experimentation, taken together, constitute computational reflection. [18, 19] That is, the SoS is able to retain meta-information, reason about itself, and implement modifications to its behavior. Computational reflection is more nuanced than feedback control, but certainly less than consciousness. Importantly, we do not conceive that computational reflection will be implemented as one top-level control strategy, but will rather be distributed throughout the hierarchy of complexity of the SoS in keeping with the lessons learned from biological systems.

5 Approaches to Implementation in Mission-Critical SoS

The crucial question is, of course, how to implement this biologically-inspired approach of active experimentation coupled with computational reflection to improve and extend existing SoS, as well as to design, develop, integrate and test new ones. We suggest two complementary strategies, both of which leverage the capabilities we have listed above. One approach, which we are following in our own work, is to build testbeds [7, 12, 17] to refine our understanding of the methodologies and tools required to incorporate active experimentation and computational reflection in a cyber-physical SoS.

The other, and more advanced, strategy is to implement portions of the required capabilities locally in an already-operating system and monitor the proposed courses of action for compatibility with known "concepts of operations" (CONOPS), which are the different styles of use intended for the system. Since the cases of most interest are also SoS providing important services that cannot be interrupted, we suggest that, after testing at the subsystem level, such modifications could be implemented during planned maintenance, update, or upgrade periods for the affected portion of the SoS. We note that all critical systems have methods for implementing such planned modifications. Addition of reflective capabilities and active experimentation could be implemented one step at a time, starting with reflection, but trapping the proposed modifications instead of implementing them. Multiple periods of testing and review could be accomplished during successive maintenance periods, carrying out all of the necessary processes for implementation except executing the proposed actions. This strategy allows a period during which the proposed actions can be compared with known CONOPS for consistency throughout the entire SoS, providing a basis for verification and validation of the expected operation of the entire SoS once the new capabilities are allowed to affect operation.

In a SoS that supplies a mission-critical service, we do not have the ability to isolate the whole system (with new incoming systems or capabilities and legacy systems) from its ongoing requirements within its true operational context. And yet, it is arguably even more critical that SoS, which are dynamic, which have many unknowns, which have constantly new combinations of legacy systems /

components and new systems / components, have some "safe" places within which to actively try out component configurations and to reason about and record / learn the impacts of such configurations in matching their requirements and operational constraints.

Most SoS are modular and utilize redundancy to achieve robustness so that sections can go down without bringing the rest of the system down, and also can be routinely taken offline for necessary check-out, maintenance, and upgrades. To leverage redundancy and maintenance periods for evaluation of the effectiveness of new capabilities such as reflection and active experimentation, one would have to devise a simulation that would mimic the current operational settings. Combining emulation / simulation and protected operation are currently done for checking out space vehicles and their subsystems and components, as well as other similarly expensive systems that require testing within very realistic operational conditions. These operational simulations could be used to test the new combinations of components, capabilities and system integrations by a human system engineer using a set of pre-designed tests. Certainly this would have great advantages over the current practices in developing and testing SoS, changing it from a certification process into one of continual verification and validation.

We now discuss our testbed, how it enables us to implement both active experimentation and computational reflection, and how we can apply what we learn to the cases of complex SoS.

6 The CARS Test Bed

CARS (Computational Architectures for Reflective Systems) is a testbed that we have been developing as an ongoing student project at California State Polytechnic University, Pomona. [7,8,17] This testbed is based on a set of design decisions that enable us to confront many of the challenges of implementing real SoS. It is composed of a group of robotic agents built from low-cost commercial off the shelf (COTS) hardware. Specifically, we use inexpensive toy radio-controlled cars and trucks. These vehicles are decidedly not ideal for the tasks we assign them, and they are also quite different one from another. Both of these circumstances mean that the self-modeling aspects of our reflective architectures are critical to successful system function. By adding our own sensors, computation, communication, and control, these toy vehicles become useful agents, although they have capabilities that are deliberately limited compared to the relatively complex tasks we require of them, a situation often replicated in real-world systems containing legacy hardware.

A series of benchmark tasks are utilized for evaluation of CARS that span a broad range of sometimes conflicting strategies: independent or multi-agent, cooperative or competitive, asynchronous or synchronous. Specifically, we use the "games" follow-the-leader, tag, soccer practice (bump a ball into a designated goal), and push-the-box (move a large, heavy object that cannot be moved by any individual agent to a designated goal). We use *Wrappings* to implement computational reflection and self-modeling. *Wrappings* grew out of work on conceptual

design environments for space systems, and has been in continuous development since its inception in 1989. [4, 13–15]

Some of the important characteristics of CARS are

- The cost of each robotic motion platform (< $50) means that, unlike most deployed systems, the investment in any part of the system is relatively small. (A new agent can be prepared in less than a day from COTS hardware and the electronics of a damaged agent.)
- The robotic components are relatively crude, requiring more modeling and self-refinement of generic models than better hardware.
- The performance of the SoS for any task can be evaluated from recorded video of the "field of play."
- The tasks and the appropriate performance measures are easy to express in everyday language.
- Use of *Wrappings* frees experimenters from many of the detailed programming tasks normally associated with adding or modifying a process, model, or sensor interface.

What Wrappings provides here is the ability for the system to have multiple alternative resources for any given problem, and to select them according to their operational context at the time of use. Because the process that make those selections are also resources, and are also selected just like any other, these systems have a very strong kind of computational reflection [15, 16]. The Wrappings approach also allows active experimentation in two ways. First, the system can create or otherwise collect new resources and try them out in a context that indicates simulation and evaluation, thus not needing to activate them in the "real" operational system until they are deemed to be ready. Second, the system can adjust the context conditions under which certain resources are selected and adapted, so that resources may be used in different ways.

We now speculate on the applicability of both the CARS testbed and the incremental approach as strategies for eventually implementing active experimentation and computational reflection in mission-critical SoS.

7 Prospects

In the CARS testbed, we have the luxury of allowing the system and its agents in the true operational environment to practice, make mistakes, learn its characteristics (e.g., turning ratio, speed on different surfaces etc.), and even damage an agent without dire consequences to itself or to the rest of the testbed, somewhat as children learn their capabilities and the constraints of their various environments through play. However, in addition to pre-defined test sets, we speculate that in fact the style of self-modeling, learning, and subsequent recording of new rules and constraints that we have advocated for the CARS testbed could become very useful for offline testing and progressive integration of parts of a SoS. In our approach, each component and subsystem of the CARS is constantly developing better and better rules and constraints on its behavior and its allowable

operational envelope. The result is that because the "experimentation" is being developed in parallel from the point of view of many different types of components playing their diverse roles, the system is very likely to discover much more about potential problems than a test set developed by even a knowledgeable and experienced system engineering team.

This kind of exploratory behavior is an extension of exploring the system's external environment to exploring the space of potential behaviors. Since this space is enormous, some very powerful directive constraint mechanisms will be needed to keep the system within some reasonable expectations, and some very powerful verification and validation methods will be needed to assure us that the system will accede to any safety- and mission- critical constraints we may choose to impose.

Eventually, we can envision a situation in which the components themselves when faced with a novel component interface or configuration or operational setting can request a time out, a voluntary removal of themselves to maintenance / self-examination / hypothesis generation and testing mode in a simulation. Imagine that in addition to the meta-knowledge normally provided to a SoS broker, each component / system has strong self-models at multiple time and space resolutions that are being continually refined with interaction with other components and environments. Initially, as a new configuration of components is brought together with the top level descriptions provided to the broker in the Wrappings, there will now be a deeper process of negotiation among the components as their self-models now compare constraints, expectatins, rules for best practice, and other behavior modification and constraint conditions. If a component is now faced with either an unknown situation (a new condition for which it has no rules or constraints) or a partially violated constraint (whose priority might not be that high), it can request that the system allow it to temporarily go into maintenance mode.

Of course, to be able to entertain this type of negotiation will take more information in the self-model about that component's expected CONOPS, in addition to its expected environment. The system will either have some type of holding action it can take or it might request the broker to provide it a new component and go on. Meanwhile the offline component now starts a set of experiments in the safe simulation, with current operational setting values and conditions and with either the other relevant software components (clones) and / or emulated hardware components. If its experiments go well enough (measured by seriousness of system use), then that component can go out of maintenance mode and back online. At that point, it tracks and records all the real results of its interactions in this new use or configuration for future rules and constraints. If the results of the experimentation are equivocal, then human intervention may be requested for further experiments.

To summarize then, we want to develop methods that allow and even encourage processes that continually improve the performance of a system through better use of its existing resources, the correct incorporation of new component

resources, and the most appropriate integration among resources given the current operational context and CONOPS.

8 Conclusion

A study of biological systems suggests possible strategies for creating robust adaptive responses of a complex SoS to changing or even unanticipated conditions. In this paper, we focused on one such strategy: active experimentation. We have shown that successful active experimentation requires several specialized capabilities that, taken together, amount to computational reflection. We then proposed various strategies for implementing active experimentation and computational reflection in mission-critical systems of systems.

We have suggested that having testbeds like CARS allows components, subsystems, and systems to build ever-improving self-models based on active experimentation. The active experimentation coupled with the reflective reasoning processes allows these components / systems to develop and refine rules and constraints with specific details about different operating conditions and other components or systems.

We have then speculated that some of these approaches could be applied to mission-critical SoS by taking advantage of the offline maintenance mode allowed for most SoS components / subsystems. This second best case is to have during maintenance, some way of setting up a safe operational environment (set of simulations and emulations) for the offline components to actively experiment performing new behaviors, joining in novel configurations of components, or experiencing new operational settings. This experimentation would help refine the current self-models to take into account these new conditions.

The last case is to develop a new style of negotiation where components are outfitted not only with their own constraints and behavioral rules, but also CONOPS that helps explicitly define the expectations for how this component is expected to be used under different circumstances. This negotiation would be going on in parallel across layers of systems and components, allowing many lines and types of detailed interactions to be analyzed by the self-modeling processes. In this last most speculative case, based on this negotiation, individual components would request being put into study mode (offline maintenance mode and into operational simulation mode) in order to follow up on any conflicts with current constraints or lack of information on requested behaviors.

We are hoping this paper will stimulate a community wide discussion into many different ways one could create safe places for self-modeling and experimentation resulting in better system integration, system validation, and system performance.

References

1. Tegest Aychek and Steffen Jung. The axis of tolerance. *Science*, 343(6178):1439–1440, 2014.

2. Michael Bell. Introduction to service-oriented modeling. In *Service-Oriented Modeling: Service Analysis, Design, and Architecture*. Wiley, 2008.
3. Kirstie Bellman, Christopher Landauer, and Phyllis Nelson. Systems engineering for organic computing: The challenge of shared design and control between oc systems and their human engineers. In Rolf Würtz, editor, *Organic Computing*, volume 21 of *Understanding Complex Systems*, pages 25–80. Springer Berlin/Heidelberg, 2008.
4. Kirstie L. Bellman, April Gillam, and Christopher Landauer. Challenges for conceptual design environments: The vehicles experience. *Revue Internationale de CFAO et d'Infographie*, September 1993.
5. Kirstie L. Bellman and Christopher Landauer. Computational embodiment: Biological considerations. *Proceedings of ISAS'97: The 1997 International Conference on Intelligent Systems and Semiotics: A Learning Perspective*, pages 422–427, 1997.
6. Kirstie L. Bellman and Christopher Landauer. Reflection processes help integrate simultaneous self-optimization processes. In *Proceedings Second International Workshp on Self-Optimization in Organic and Autonomic Computing Systems (SAOS 2014), 27th International Conference on Architecture of Computing Systems (ARCS 2014)*. Lübeck, Germany, February 2014.
7. Kirstie L. Bellman, Christopher Landauer, and Phylis R. Nelson. Managing variable and cooperative time behavior. In *First IEEE Workshop on Self-Organizing Real-Time Systems*, Carmona, Spain, May 2010.
8. Kirstie L. Bellman and Phyllis R. Nelson. Developing mechanisms for determining 'good enough' in sort systems. In *2nd IEEE Workshop on Self-Organizing Real Time Systems (SORT 2011)*, Newport Beach, CA, March 2011. (presentation).
9. Kirstie L. Bellman and Donald O. Walter. Biological processing. *American Journal of Physiology*, 246:R860–R867, 1984.
10. M. Kwiatkowska. Advances in quantitative verification for ubiquitous computing. In *Proc. 11th International Colloquium on Theoretical Aspects of Computing (ICTAC 2013)*, volume 8049 of *LNCS*, pages 42–58. Springer, Heidelberg, 2013.
11. M. Kwiatkowska, D. Parker, H. Qu, and M. Ujma. On incremental quantitative verification for probabilistic systems. In Andrei Voronkov and Margarita Korovina, editors, *HOWARD-60: A Festschrift on the Occasion of Howard Barringer's 60th Birthday*, pages 245–25. 2014.
12. Christopher Landauer. Abstract infrastructure for real systems: Reflection and autonomy in real time. In *Proceedings SORT 2011: The Second IEEE Workshop on Self-Organizing Real-Time Systems*, Newport Beach, California, March 2011. (presentation).
13. Christopher Landauer. Infrastructure for studying infrastructure. In *Proceedings of ESOS 2013: Workshop on Embedded Self-Organizing Systems*, San Jose, California, June 2013. (presentation).
14. Christopher Landauer and Kirstie L. Bellman. Generic programming, partial evaluation, and a new programming paradigm. In Gene McGuire, editor, *Software Process Improvement*, chapter 8, pages 108–154. Idea Group Publishing, 1999.
15. Christopher Landauer and Kirstie L. Bellman. Self-modeling systems. In H. Shrobe R. Laddaga, editor, *Self-Adaptive Software*, volume 2614 of *Springer Lecture Notes in Computer Science*, pages 238–256. Springer Berlin/Heidelberg, 2002.
16. Christopher Landauer and Kirstie L. Bellman. Managing self-modeling systems. In H. Shrobe R. Laddaga, editor, *Proceedings of the Third International Workshop on Self-Adaptive Software*, Arlington, Virginia, June 2003.

17. Christopher Landauer, Kirstie L. Bellman, and Phyllis R. Nelson. Modeling spaces for real-time embedded systems. In *Proceedings SORT 2013: The Fourth IEEE Workshop on Self-Organizing Real-Time Systems*, Paderborn, Germany, June 2013. presentation.
18. P. Maes and D. Nardi, editors. *Meta-Level Architectures and Reflection*. North Holland, 1988.
19. Pattie Maes. Computational reflection. In Katharina Morik, editor, *GWAI-87 11th German Workshop on Artifical Intelligence*, volume 152 of *Informatik-Fachberichte*, pages 251–265. Springer Berlin Heidelberg, 1987.
20. Arthur Mortha, Aleksey Chudnovskiy, Daigo Hashimoto, Milena Bogunovic, Sean P. Spencer, Yasmine Belkaid, and Miriam Merad. Microbiota-dependent crosstalk between macrophages and ilc3 promotes intestinal homeostasis. *Science*, 343(6178):1477, 2014.
21. David J. Musliner, Timothy Woods, and John Marais. Identifying culprits when probabilistic verification fails. In *Proc. ASME Computers and Information in Engineering Conference*. August 2012.
22. M. Ryschkewitsch. Engineering of complex systems: Challenges and initiatives. In *7th Annual IEEE Systems Conference (SysCon)*, Orlando, FL, 2013.
23. W.T. Tsai, Xinyu Zhou, Yinong Chen, Bingnan Xiao, R.A. Paul, and W. Chu. Roadmap to a full service broker in service-oriented architecture. In *IEEE International Conference on e-Business Engineering (ICEBE 2007)*, pages 657–660. October 2007.
24. Serdar Uckum, Tolga Kurtoglu, Peter Bunus, Irem Tumer, Christopher Hoyle, and David Musliner. Model-based systems engineering for the design and development of complex aerospace systems. In *SAE Aerotech*. 2011.
25. Paolo Zuliani, Andr Platzer, and Edmund M. Clarke. Bayesian statistical model checking with application to stateflow/simulink verification. In *Formal Methods in System Design*, volume 43, pages 338–367. Springer, 2013.

Formal Framework for Ensuring Consistent System and Component Theories in the Design of Small Satellite Systems

Jules Chenou[1], William Edmonson[1], Albert Esterline[1], and Natasha Neogi[2]

[1] NC A & T State University
Greensboro, NC 27411
Email: {jchenou,wwedmons,esterlin}@ncat.edu
[2] National Institute of Aerospace
Hampton, VA 23666
Email: neogi@nianet.org

Abstract. We present a design framework for small-satellite systems that ensures that (1) each satellite has a consistent theory to infer new information from information it perceives and (2) the theory for the entire system is consistent so that a satellite can infer new information from information communicated to it. This research contributes to our Reliable and Formal Design (RFD) process, which strives for designs that are "correct by construction" by introducing formal methods early. Our framework uses Barwise's channel theory, founded on category theory, and allied work in situation semantics and situation theory. Each satellite has a "classification", which consists of tokens (e.g., observed situations) and types (e.g., situation features) and a binary relation classifying tokens with types. The core of a system of classifications is a category-theoretic construct that amalgamates the several classifications. We show how to derive the theory associated with a classification and the theory of the system core, and we show how to check whether a given requirement is derivable from or consistent with a theory.

1 Introduction

This research represents our ongoing effort to develop a model-based systems engineering methodology for small satellite systems that is reliable, formal and results in a "correct-by-construction" design. We present a knowledge-based design framework for ensuring that (1) each satellite has a consistent theory it can use to infer new information from information it perceives and (2) the theory for the entire system is consistent so that a satellite can infer new information from information communicated to it, and it can assess purported information from fellow satellites. The theories mentioned can be updated as insight is gained about the sensing satellites and the part of the world observed.

This research contributes to our development of a Reliable and Formal Design (RFD) process [1] [2], which strives for designs that are correct by construction by introducing formal methods early in the design process so that redesign may

be minimized. In previous work, we defined the structure for checking consistency and traceability of requirements in a formal manner [2]. In this paper, we expand on our definition of consistency in terms of category theory and information flow as it relates to designing systems of heterogeneous satellites that share information about the situations they observe.

Consider a particular small satellite S. A type (for S) is a feature of interest that S may observe in a monitored situation. We assume that we can identify a set of types that characterize S's sensory capabilities and, further, that we can form a table indicating the combinations of types that may occur together in an observed situation. In Section IV.A, we present an algorithm to derive a theory (for S) from this classification table. The algorithm can detect inconsistency. We may have to update S's classification table: we may find a new combination of types that may occur together (and so add a row to the table), or we may find that a previously postulated combination is in fact spurious (and so delete a row from the table). We may even find it necessary to introduce a new type for S. Whenever S's classification table is updated, we can run our algorithm and update S's theory (and check for consistency). S's theory allows it to infer additional information about a situation from information about that situation that it has perceived or received via communication with other satellites. S's theory also allows it to detect when purported information about a given situation from another satellite is incoherent from its (that is, S's) point of view or is inconsistent with the information S has about the situation in question.

We are concerned with a system of small satellites, so we consider the amalgamation of the classification tables of the several satellites in a given system as well as the theory inherent in this amalgamation. The amalgamation combines the content of the classification tables for the individual satellites and adds relations among types from several satellites. We describe this amalgamation in Section IV.C in terms of category theory. We can think of the system (with its classification table and theory) as the "whole" and the individual satellites as "parts". Again, in deriving the theory, we can determine whether it is consistent, and we can update the theory of the whole when the classification table for a part or a cross-part type dependency changes.

We describe, in an abstract manner, the algorithms and procedures that are used at a system-wide level in order to manipulate information and general knowledge for the satellite system. We do not suggest that the final implementation of the system of satellites will use these procedures. Rather, they are to be taken in the spirit of executable specifications. Later stages of the design process will refine the specifications into concrete designs that can be directly implemented.

The higher-level view taken here is based in the first instance on category theory, which captures abstract algebraic structures and their interactions in a coherent way (as categories) and also captures the relations between the categories. We make use of Barwise and Seligman's channel theory [3], which is an application of category theory to the "flow of information". Information is addressed here, not in terms of the amount of information as per the discipline

initiated by Shannon, but in the sense of the information content of an event or message. Flow here is not to be interpreted in a physical sense, but in the sense that X being α carries the information that Y is β.

Barwise and Seligman developed channel theory to explain the notion of a constraint, which was central in Barwise's account of "X being α carries the information that Y is β" in situation semantics. Barwise and Perry [4] developed situation semantics as a general theory of naturalized meaning and information. Devlin [5] provided a systematic presentation of situation semantics and the situation theory behind it. We exploit situation semantics and situation theory along with channel theory to give a rigorous and general account of the information satellites in a system of satellites have and share.

A *situation* is an ongoing happening in the world, while an *infon* is the basic item of information. An infon involves an n-place relation R, n objects appropriate for the corresponding argument places of R, a spatiotemporal location, and a polarity of 1 (indicating that the objects are thus related at the indicated place and time) or 0 (indicating otherwise). Alternatively, one may think of a situation as a partial function from tuples, with all the above components except polarity, to the codomain $\{0, 1\}$. A situation, unlike a possible world in the semantics of modal logics, targets only part of the world. A *real situation* is a part of reality (and considered a single entity) that supports an indefinite number of infons, while an *abstract situation* is a fixed set of infons.

Information flow is made possible by uniformities across relations between situations, that is, (as in channel theory) *constraints* (including natural laws and linguistic rules) that link various types of situations. Situation semantics addresses speech acts (utterances) and the "situatedness" of language use. It presents a theory of meaning that is relational in that language use relates situations: a linguistic unit (such as a declarative sentence) is uttered in an "utterance situation" whose descriptive content is some "described situation".

We retain the terminology of situation semantics for communication among small satellites since viewing satellite communication at a high level, in terms of utterance situations, allows us to abstract away unnecessary detail that adds nothing to the analysis. It also allows us to exploit notions relating to conversation that are not applicable at a lower level. A satellite broadcasting information delineates a spatiotemporal region in which there occurs an utterance situation (which also includes the satellites that are its physical neighbors given that an appropriate channel exists). We are interested in the information, its flow, conventions for taking turns, assumptions about content, and many other things at the level of speech and conversation. We are not interested in the physical aspects of communication and so eschew terms such as "broadcast".

The information of interest here concerns only the observed situations. The types relate only to observed situations, never to the satellites themselves. We do not address the issue of control actions taking by the satellites nor are we concerned with how information is extracted from signal. Real situations, with real objects and happenings, are observed although different satellites observe the same real situation from different perspectives and with different modalities.

We generally prefer "observe" for the verb relating to the source of information unless the emphasis is on the medium, in which case we tend to use "perceive".

We view the critical aspects of a small satellite as an intelligent knowledge based system (IKBS). An IKBS as proposed here has a syntactic aspect and a semantic aspect. The semantic aspect relates to the specific kind of information that the satellite can process and is represented by a finite number of "classifications." For generality, we allow for several classifications associated with the same satellite since it is common for a single satellite to have several sensing modalities or to use several fusion techniques. Each classification $\mathcal{A}$ (in the sense of channel theory) consists of a set $typ(A)$ of types and a set $tok(A)$ of tokens; a token $a \in tok(A)$ may be classified of type $\alpha \in typ(A)$ in classification A, written $a \vDash_A \alpha$ and called a *simple proposition*. So a classification $\mathcal{A}$ is a triple, $(tok(A), typ(A), \vDash_A)$. For a small satellite system, tokens are real situations (as observed by the satellites). All the classifications with which a given satellite is endowed are amalgamated into a core classification (as explained in Section IV) for that satellite, which we shall refer to as the classification of the IKBS.

The syntactic aspect of the IKBS of a satellite is a set of implication rules, which are the constraints mentioned above. Each rule is a sequent, of the form $\Gamma \vdash \Delta$, where Γ and Δ are sets of types. Suppose the classification in question here is $\mathcal{A}$. This sequent is *satisfied* by a token (real situation) a as long as, if $a \vDash_A \alpha$ for all $\alpha \in \Gamma$, then $a \vDash_A \beta$ for some $\beta \in \Delta$. A sequent is a constraint (for $\mathcal{A}$) if it is satisfied by all tokens in $tok(\mathcal{A})$. The deductive closure of the set of constraints is the IKBS's (or satellite's) theory (discussed above). The deductive closure of the set of rules is the IKBS's (or satellite's) theory (discussed above). Such a rule, if appropriately enabled, allows the satellite to infer new infons from its IKBS given other infons in the observed situation or the described situation associated with an utterance situation (where the utterance is by a neighboring satellite).

The next section describes our Reliable and Formal Design (RFD) process and how the techniques reported in this paper fit into this framework. Section III presents enough category theory and channel theory so that the reader may understand the rest of this paper. Section IV is the technical heart of this paper and presents techniques for amalgamating classifications, deriving a theory from a classification, and checking whether a requirement (encoded as a sequent) is derivable from a theory or may be consistently added to it. Section V considers how the satellites in a system maintain and communicate information on real situations. We assume that an IKBS may use its theory or the theory for the entire system to infer new information. Special attention is given to communication since an IKBS may fail to interpret an utterance or the content of what is uttered may be inconsistent with the information the IKBS has. Section VI concludes.

2 Reliable and Formal Design Process

The RFD (Reliable and Formal Design) process [2] follows a risk-tolerant philosophy that notionally will lead to a correct design with minimal-to-no redesign through the use of an agile and formal design process based on models. By integrating formal methods into the proposed design process at the appropriate levels, many design failures and integration challenges can be eliminated. Formal methods will provide automatic means for verification by translating requirements into a higher order logic language for which tools such as PVS [6] can perform consistency and traceability checks and proofs throughout the design process.

This RFD systems engineering process takes into account the fact that the design team is small and multi-disciplinary as well as the fact that system is complex and heterogeneous and is affected by its operational environment. Additionally, design of a complex system involves multiple disciplines that must interact symbiotically. This also implies that at the first step in the process each of the disciplines must have a global view of the system. As they proceed towards refinement, the design process becomes a local or discipline-specific activity, though always with a global perspective. The integration of formal methods into the design process of a complex system can reduce the need for a significant amount of revision during the system integration phase because of the "correct by construction" nature of the process. This leads to testing virtually at each level of refinement. Therefor, the theme of this design process from a systems engineering point of view is: *Think globally, design locally, and test virtually.*

At each level of abstraction, $\mathcal{A}_i$, the state of the RFD process can be represented by requirements, models, and simulations: $\mathcal{A}^i = (\mathcal{L}_n^i, \mathcal{L}_l^i, \mathcal{M}^i, \mathcal{S}_p^i, \mathcal{S}_b^i)$,where the information flow between these components is indicated with arrows as follows:

$$\begin{array}{ccccccc} \mathcal{L}_n^i & \Longleftrightarrow & \mathcal{L}_l^i & \Longleftrightarrow & \mathcal{M}^i & \Longrightarrow & \mathcal{S}_p^i \\ & & \Downarrow & & & & \\ & & \mathcal{S}_b^i & & & & \end{array} \tag{1}$$

Here

- $\mathcal{L}_n^i$ is the set of requirements written in natural language form
- $\mathcal{L}_l^i$ is the set of requirements written as a set of logical functions
- $\mathcal{M}^i$ is the system of interconnected models
- $\mathcal{S}_p^i$ is the set of simulations based on the parameters of $\mathcal{M}^i$.
- $\mathcal{S}_b^i$ is the set of simulations based on the logical description in $\mathcal{L}_l^i$.

The central result of this paper is the technique presented in Section IV.A for developing a theory for a satellite/IKBS from the classification table for it. As this addresses individual satellites, it is in the realm of local design, albeit at a very abstract level. The classification system used by an IKBS may be an amalgamation of several classifications. We form the classification for the entire

satellite system by a similar amalgamation (see Section IV.C) given the classifications of the component IKBS; the theory for the system is derived from this classification table. This is a return to a global perspective since the individual satellites with their sensing and communication capabilities were originally selected or built as required by the global mission. Theories for individual satellites and for the entire system are tested for consistency, which is virtual testing from the point of view of the system being developed.

The aspect in this knowledge-level design process emphasized in this paper is endowing each IKBS present within the distributed system with a theory in the form of a set of constraints. One determines for each IKBS the core classification with which it should be endowed and the classifications into which this core should be decomposed as "parts". Henceforth, we call the core classification for the IKBS simply the "IKBS classification". One then determines the types for each of the "part" classifications and the IKBS classification. These are the features of the IKBS's environment that it can sense or be informed about. One then produces a *classification table* for each "part" and for the core. The columns in the classification table are labeled with the classification's types, and we have a row for each abstract situation that (to our knowledge) may arise for that classification. A row has an '1' under a type if the abstract situation represented by that row supports that type; otherwise, it has a '0' under the type. Once the classification table for the IKBS classification is fixed, one can determine the IKBS's initial theory (a set of constraints) as described in Section IV.A. Determining the classifications for an IKBS can be done in a top-down fashion by first determining its core (or IKBS) classification and then determining the relative parts in a way that is sensitive to its physical attributes. Alternatively, this can be done in a bottom-up fashion, by mixing and matching basic classifications in an engineering repertoire and forming the IKBS classification once the parts are given.

Given a system of small satellites, we consider the amalgamation (the core) of the IKBS classifications. As with forming an IKBS classification, forming the core for the entire system is (in category-theory terms) a sum operation. Section IV.C shows how the classification table for the system core is constructed from the classification tables for the individual IKBSs. Again, we can form the initial theory of the system using the techniques described in Section IV.A. Combining theories raises the threat of inconsistency, and we check for inconsistency syntactically by determining whether, from the combined theory, we can derive some formula and its negation or, equivalently, we can derive the empty sequent $\langle \emptyset, \emptyset \rangle$. Consistency can also be defined semantically: see Section IV.B.

Designing a small satellite system is not just a matter of selecting the right kinds of satellites, developing their theories, and combining their reports at the knowledge level. One must also determine the overall situation, extended in both space and time, that the system will perceive. And one must formulate policies for how this overall situation will be divided into overlapping component situations allocated as regions of responsibility to the different satellites; these are policies for allocating sequences of overlapping situations to the various satellites. How

regions of responsibility should overlap depends both on what is monitored and the capabilities and theories with which the satellites are endowed. Significant overlap is expected when two satellites have complementary sensing modes or perspectives on a target.

How a satellite is actually deployed may give rise to additional constraints to be incorporated into its theory. And how several satellites are designed to collaborate may affect how the system core is formed and thus the theory for the system whole.

Once the satellite system is deployed, constraints will be added to the IKBSs, and their theories as well as the theory of the system as a whole will evolve. That is, new abstract situations will become apparent and could be added to the appropriate classification tables. One can check that the resulting theory remains consistent. One of the strengths of our approach is that, at any point in the lifetime of the systems, new requirements can be checked for consistency against current theories and, when consistent, used to augment those theories. At any time, the various theories must meet their requirements that have been collected over time. One way a theory can violate a requirement is by failing to include it. The solution in that case is to add the requirement to the theory and test for consistency. Another way a theory can violate a requirement is to be inconsistent with it, in which case the theory must be adjusted.

3 Category Theory and Channel Theory Overview

This section provides background for formal addressing the aggregation and flow of information in a small satellite system. The topics discussed here include category theory, channel theory, and the application of these theories to computational systems.

3.1 Category Theory

A category C consists of a class of objects and a class of morphisms (or arrows or maps) between the objects. Each morphism f has a unique source object a and target object b; we write$f : a \to b$. The composition of $f : a \to b$ and $g : b \to c$ is written as $g \circ f$ and is required to be associative: if in addition $h : c \to d$, then $h \circ (g \circ f) = (h \circ g) \circ f$. It is also required that, for every object x, there exists a morphism $1_x : x \to x$ (the identity morphism for x) such that, for every morphism $f : a \to b$, we have $1_b \circ f = f = f \circ 1_a$. It follows from these properties that there is exactly one identity morphism for every object.

A functor from one category to another is a structure-preserving mapping, preserving the identity and composition of morphisms. More exactly, if C and D are categories, then a functor F from C to D is a mapping that associates with each object $x \in C$ an object $F(x) \in D$ and, with each morphism $f : x \to y \in C$, a morphism $F(f) : F(x) \to F(y) \in D$. In addition, it requires that $F(id_x) = id_{Fx}$ for every object $x \in C$, and $F(g \circ f) = F(g) \circ F(f)$ for all morphisms $f : x \to y$ and $g : y \to z$. In category theory, a commutative diagram is a diagram of objects

(as vertices) and morphisms (arrows between objects) such that all directed paths in the diagram with the same start and end points lead to the same result by composition.

The classic presentation of category theory is in [6]. Two reasonably comprehensive and rigorous texts that are accessible to most readers with classical engineering mathematical backgrounds are [7] and [8]. A light introduction is provided by [9], while [10] and [11] are category-theory texts addressed specifically to computer scientists; [12] addresses category theory in the context of software engineering.

3.2 Channel Theory and the Flow of Information

Barwise and Seligman [3] presented a framework for the "flow of information" in (generally implicit) category-theoretic terms. They address the question, "How is it that information about any component of a system carries information about other components of the system?" For classifications $\mathcal{A}$ and $\mathcal{C}$, an infomorphism f from $\mathcal{A}$ to $\mathcal{C}$ is a pair of functions $(f^\wedge, f^\vee)$, $f^\wedge : typ(\mathcal{A}) \rightarrow typ(\mathcal{C})$, and $f^\vee : tok(\mathcal{C}) \rightarrow tok(\mathcal{A})$, satisfying, for all tokens $c \in tok(\mathcal{C})$ and all types $\alpha \in typ(\mathcal{A})$, $f^\vee(c) \models_{\mathcal{A}} \alpha$ iff $c \models_{\mathcal{C}} f^\wedge(\alpha)$. In category theoretic terms, an informorphism is a kind of *Galois connection*. A *contravariant* Galois connection between $(tok(\mathcal{A}), typ(\mathcal{A}), \models_{\mathcal{A}})$ and $(tok(\mathcal{B}), typ(\mathcal{B}), \models_{\mathcal{B}})$ will be a pair (φ, ψ) of mappings $\varphi : tok(\mathcal{A}) \longrightarrow typ(\mathcal{B})$, $\psi : tok(\mathcal{B}) \longrightarrow typ(\mathcal{A})$ satisfying $a \models_{\mathcal{A}} \psi(b)$ if and only if $b \models_{\mathcal{B}} \varphi(a)$. An infomorphism between $(tok(\mathcal{A}), typ(\mathcal{A}), \models_{\mathcal{A}})$ and $(tok(\mathcal{B}), typ(\mathcal{B}), \models_{\mathcal{B}})$ however is a *covariant* Galois connection. It is a contravariant Galois connection between $(tok(\mathcal{A}), typ(\mathcal{A}), \models_{\mathcal{A}})$ and $(typ(\mathcal{B}), tok(\mathcal{B}), \models_{\mathcal{B}}^{-1})$ where $\models_{\mathcal{B}}^{-1} \subseteq typ(\mathcal{B}) \times tok(\mathcal{B})$ with $(\beta, b) \in \models_{\mathcal{B}}^{-1}$ if and only if $(b, \beta) \in \models_{\mathcal{B}}$.

Components of the system may be distant from one another in time and space, and the system can be made up of heterogeneous components. The system is "distributed" in this sense and not necessarily in the classical sense used in computer science. For example, the students, classrooms, scheduling system, and attendance records together form a distributed system related to students' attendance at a certain university.

An *information channel* is a family of infomorphisms with a common codomain, called the *core*. Essentially, a channel consists of a set $\mathcal{A}_1, ..., \mathcal{A}_n$ of classifications that represent the parts of the distributed system, a classification $\mathcal{C}$ (the core) that represents the system as a whole, and a set of infomorphisms $f_1, ..., f_n$ from each of the parts onto $\mathcal{C}$. Tokens in $\mathcal{C}$ are the *connections* of the system: a given token c in $\mathcal{C}$ connects the tokens it is related to by means of $f_1, ..., f_n$. Parts $\mathcal{A}_1, ..., \mathcal{A}_n$ carry information about each other as long as they all are parts of $\mathcal{C}$. Intuitively, an information channel is a part to whole $\mathcal{A}_i$-to-$\mathcal{C}$ informational relationship. Categorically, the core is a cocone in the category of classifications (objects are classifications and morphisms are informorphisms).

A *distributed system* D is a collection of elements that carry information about each other. Formally, D consists of an indexed class $cla(D)$ of classifications together with a class of infomorphisms, $inf(D)$, whose domains and

codomains are all in $cla(D)$. An information channel C covers distributed system D if and only if $cla(D)$ are the classifications of the channel and, for every infomorphism $f \in inf(D)$, there are infomorphisms from both the domain and codomain of f to the core of C such that these three infomorphisms are commutative in there interrelation. Basically, all classifications in D are "informational parts" of the core whose channel covers D.

Turning to regularities in a classification's types, let $\mathcal{A}$ be a classification and Γ and Δ be subsets of types in $\mathcal{A}$. Recall (from Section I) that a token a of $\mathcal{A}$ satisfies the *sequent* $\Gamma \vdash \Delta$ provided that, if a is a token of every type in Γ, then it is of some type in Δ. If every token of $\mathcal{A}$ satisfies $\Gamma \vdash \Delta$, then Γ is said to entail Δ and $\Gamma \vdash \Delta$ is called a *constraint* supported by $\mathcal{A}$. The set of all constraints supported by A is called the complete theory of $\mathcal{A}$, denoted by $Th(\mathcal{A})$. These constraints are systematic regularities, and it is by virtue of regularities among connections that information about certain component of a distributed system can be carried by other components into diverse parts of the system. Barwise and Seligman's summary statement of their analysis of information flow, restricted to the simple case of a system with two components, is as follows. "Suppose that the token a is of type α. Then a's being of type α carries the information that b is of type β, relative to channel C, if a and b are connected in C and if the translation β' of β entails the translation α' of α in the theory $Th(C)$, where $\mathcal{C}$ is the core of C" ([3], p.35).

3.3 Category Theory and Channel Theory for Computational Systems

Channel theory and category theory in general have had a significant impact in computer science, especially those aspects related to complex systems. Schorlemmer and Kalfoglou and their colleagues have applied channel theory in addressing semantic interoperability of federated databases [13] as well as the similar problem of ontology alignment [14] . Kent's Information Flow Framework (IFF) [15] also uses channel theory; IFF is being developed by the IEEE Standard Upper Ontology working group as a meta-level foundation for the development of upper ontologies. Spivak [16] presents a simple database definition language based on categories and functors and shows how to translate instances from one database schema to the other in canonical ways; Spivak and Kent [17] have also defined a category-theoretic model, OLOG, for knowledge representation. Finally, Diskin and Maibaum [18] support the claim that category theory provides a toolbox of design patterns and structural principles of real practical value for model driven software engineering. These research programs are generally at the knowledge level and as such fit well with the work reported here.

Goguen and Burstall's theory of institutions [19] is a categorical abstract model theory that formalizes the intuitive notion of a logical system, including syntax, semantics, and the satisfaction relation between them, which relates a semantic model to syntactically well-formed formulas that can be interpreted as true given the model. The meaning of the satisfaction condition of institutions is that truth is invariant under change of notation. Goguen has used institutions as

a basis for unifying and generalizing several approaches to information, including channel theory, Formal Concept Analysis, and Sowa's lattice of theories (ordered by inclusion on sets of derivable formulas) used for knowledge representation [20]. The work reported here gains impact by fitting into the general framework provided by the theory of institutions.

4 Amalgamating Classifications, Deriving Theories, and Checking Consistency

Figure 1 depicts the aspect of the RFD process presented here. On the left-hand side of the top, we have a specification in a natural language, $\mathcal{L}_n$, from which classification tables in the logical language $\mathcal{L}_l$ are encoded. From these tables, one generates a set of sequents (or constraints) using the algorithm presented in this section. The deductive closure of this set is the IKBS theory. Note that a theory is inconsistent if and only if one can derive the empty sequent, $\emptyset \vdash \emptyset$ from it. On the right-hand side of Figure 1, we have a requirement expressed in natural language $\mathcal{L}_n$. This is encoded as a sequent in $\mathcal{L}_l$. One then tests whether the sequent is entailed by the theory, is inconsistent with the theory (i.e., the theory with the sequent added is inconsistent), or neither. If it is entailed by the theory, we have increased confidence in both the theory and the requirement. If the requirement is inconsistent with the theory, then both the requirement and the theory could be suspect. If the requirement survives scrutiny, one redesigns the theory. Redesign in this approach focuses on the encoding and the correctness of the original $\mathcal{L}_n$ statements. If the sequent that encodes the requirement is not entailed by nor inconsistent with the theory, then one adds the sequent to the set of sequents whose deductive closure forms the theory.

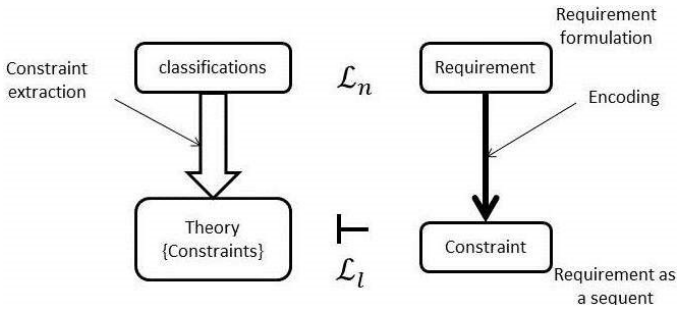


Fig. 1. RFD process as presented here.

If $\mathcal{A} = (tok(\mathcal{A}), typ(\mathcal{A}), \models_{\mathcal{A}})$ is a classification, we define

$$\alpha' = \{a \in tok(\mathcal{A}) | a \models_A \alpha\} \tag{2}$$

$$\Gamma^{\cap} = \Gamma' = \{x \in tok(\mathcal{A}) | \forall \alpha \in \Gamma, x \models_A \alpha\} = \bigcap_{\alpha \in \Gamma} \alpha' \tag{3}$$

$$\Gamma^{\cup} = \{x \in tok(\mathcal{A}) | \exists \alpha \in \Gamma, x \models_A \alpha\} = \bigcup_{\alpha \in \Gamma} \alpha' \tag{4}$$

It is straightforward that a token x satisfies the sequent $\langle \Gamma, \Delta \rangle$ if $x \in \Gamma^{\cap} \Rightarrow x \in \Delta^{\cup}$. Thus,$\langle \Gamma, \Delta \rangle$ is a constraint of $\mathcal{A}$ if and only if $\Gamma^{\cap} \subseteq \Delta^{\cup}$. An *implication* $\Gamma \rightarrow \Delta$ between types in $\mathcal{A}$ is a is sequent $\langle \Gamma, \Delta \rangle$ where Δ is non-empty. A subset $T \subseteq typ(\mathcal{A})$ respects an implication $\Gamma \rightarrow \Delta$ if $\Gamma \nsubseteq T$ or $\Delta \subseteq T$. An *implication* $\Gamma \rightarrow \Delta$ holds in $\mathcal{A}$ if and only if every subset of $typ(\mathcal{A})$ respects $\Gamma \rightarrow \Delta$ and so,if and only if $\Gamma^{\cap} \subseteq \Delta^{\cap}$

The Duquenne–Guigues [21] theorem provides a way to form a canonical basis of implications from sets of types that are pseudo-closed with respect to a closure operator of a classification $\mathcal{A}$. A set Γ of types is *pseudo-closed* with respect to a closure operator of a classification $\mathcal{A}$ if $\Gamma \neq (\Gamma')'$ and $(\Delta')' \subset \Gamma$ for every pseudo-closed $\Delta \subset \Gamma$. Note that in mathematics, a closure operator on a set S is a function $cl : P(S) \longrightarrow P(S)$ from the power set of S to itself which satisfies the following conditions for all sets $X, Y \subseteq S$, $X \subseteq cl(X)$, $X \subseteq Y \Rightarrow cl(X) \subseteq cl(Y)$ and $cl(cl(X)) = cl(X)$ As a base case for this definition, all minimal non-closed sets are pseudo-closed.

Algorithm 1 computes a basis of the theory of a given classification. The theory is the deductive closure of its basis. We use the Duquenne–Guigues theorem to compute part of the set of the constraints, namely, those that are implications. Note that in Algorithm 1 a sequent
$Gama \vdash \Delta$ where $\Delta = \emptyset$ is handled separately.

Algorithm 1 Constraint extraction procedure

1: **Input**: Classification $\mathcal{A}$
2: $Th(\mathcal{A}) = \varnothing$
3: For $\Gamma \subseteq typ(\mathcal{A})$
4: Compute $\Gamma^{\cup}$
5: If $\Gamma^{\cup} = tok(\mathcal{A})$ then $\Gamma \vdash \varnothing \in Th(\mathcal{A})$
6: If $\Gamma \vdash \Delta$ is in a Duquenne–Guigues basis of $\mathcal{A}$ then $\Gamma \vdash \Delta \in Th(\mathcal{A})$
7: End for
8: **Output** $Th(\mathcal{A})$

4.1 Logic of a classification

The formulas in the logic of a classification are sequents on the set of types. One can define a pseudo-negation and a pseudo-disjointion on the sequents.

A derivation or proof within this theory is as usual a sequence of sequents, where each of the sequents in the sequence is either a premise (element of the theory) or is deduced from previous sequents appearing in the sequence using one of Amstrong's rules [22], [23], described below:

Reflexivity If $\Gamma \subseteq \Delta$ then $\Gamma \vdash \Delta$

$$\frac{\Gamma \subseteq \Delta}{\Gamma \vdash \Delta}$$

Augmentation If $\Gamma \vdash \Delta$, then $\Gamma \cap \Gamma' \vdash \Delta \cup \Gamma'$

$$\frac{\Gamma \vdash \Delta}{\Gamma \cap \Gamma' \vdash \Delta \cup \Gamma'}$$

Transitivity If $\langle \Gamma, \Sigma \rangle$ and $\langle \Sigma, \Delta \rangle$ then $\Gamma \vdash \Delta$

$$\frac{\Gamma \vdash \Sigma, \Sigma \vdash \Delta}{\Gamma \vdash \Delta}$$

4.2 Consistency within a Classification Context

Consider the example classification given in Table below. We implemented Algorithm 1 in Python 2.7.5 and applied it to this classification, giving the basis of a theory as shown below.

$\vDash_A$	α	β	γ	σ
a_1	1	1	1	0
a_2	0	1	1	1
a_3	0	0	1	1

Th(A)	
$\{\alpha, \sigma\} \vdash \varnothing$	r_1
$\{\alpha, \gamma\} \vdash \{\beta\}$	r_2
$\varnothing \vdash \{\gamma\}$	r_3

Assume that a requirement of this system is expressed by the constraint $\{\alpha, \gamma\} \vdash \{\beta, \sigma\}$. We prove that this constraint is derivable from $Th(\mathcal{A})$.

1. $\{\alpha, \gamma\} \vdash \beta$ .....r_2, premise
2. $\beta \vdash \{\beta, \sigma\}\rangle$ ..... *Reflexivity*
3. $\{\alpha, \gamma\} \vdash \{\beta, \sigma\}\rangle$ .....1, 2, *Transitivity*

The proof is not always as short and easy as this. Generally, one would use an automatic or semi-automatic theorem prover such as PVS [24]. The derivability can also be shown semantically. Semantically, a sequent is valid in a classification $\mathcal{A}$ if each token of $\mathcal{A}$ satisfies the sequent. For a sequent $\Gamma \vdash \Delta$, this is equivalent to the inclusion $\Gamma^{\cap} \subseteq \Delta^{\cup}$. For example, for the sequent $\{\alpha, \gamma\} \vdash \{\beta, \sigma\}$, we have $\{\alpha, \gamma\}^{\cap} = \{a_1\} \subseteq \{a_1, a_2, a_3\} = \{\beta, \sigma\}^{\cup}$. In classical deductive logic, a consistent theory is one that does not contain a contradiction. The lack of contradiction can be defined in either semantic or syntactic terms. Here a model is taken to be a classification. The semantic definition states that a theory is consistent if and

only if it has a model, i.e., there exists an interpretation under which all formulas in the theory are true. This is the same sense used in traditional Aristotelian logic, although, in contemporary mathematical logic, the term "satisfiable" is used instead of "true". The syntactic definition states that a theory is consistent if and only if there is no formula p such that both p and its negation are provable from the axioms of the theory under the associated deductive system.

4.3 Sum of Classifications (Channel Core)

We introduced the notion of the core of an information channel in Section II.B, and we have viewed it as the whole that provides the amalgamation of the classifications of the parts. The core is formed as the category-theoretic *sum* of the classifications of the parts. Here we illustrate the sum of three classifications $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$, which is the core of the information channel containing the classifications. The sum $\mathcal{A}_1 + \mathcal{A}_2 + \mathcal{A}_3$ is the classification defined as follows:

1. The set $tok(\mathcal{A}_1 + \mathcal{A}_2 + \mathcal{A}_3)$ of tokens is the Cartesian product of $tok(\mathcal{A}_1)$, $tok(\mathcal{A}_2)$ and $tok(\mathcal{A}_3)$. Thus, the tokens of $\mathcal{A}_1 + \mathcal{A}_2 + \mathcal{A}_3$ are triples (a_1, a_2, a_3) of tokens, $a_1 \in tok(\mathcal{A}_1)$, $a_2 \in tok(\mathcal{A}_2)$ and $a_3 \in tok(\mathcal{A}_3)$.
2. The set $typ(\mathcal{A}_1 + \mathcal{A}_2 + \mathcal{A}_3)$ of types is the disjoint union of $typ(\mathcal{A}_1)$, $typ(\mathcal{A}_2)$ and $typ(\mathcal{A}_3)$. For concreteness, the types of $\mathcal{A}_1 + \mathcal{A}_2 + \mathcal{A}_3$ are pairs $\langle i, \alpha \rangle$, where $i = 1$ and $\alpha \in typ(\mathcal{A}_1)$ or $i = 2$ and $\alpha \in typ(\mathcal{A}_2)$ or $i = 3$ and $\alpha \in typ(\mathcal{A}_3)$.
3. The classification relation $\vDash_{\mathcal{A}_1+\mathcal{A}_2+\mathcal{A}_3}$ of $\mathcal{A}_1 + \mathcal{A}_2 + \mathcal{A}_3$ is defined by $(a_1, a_2, a_3) \vDash_{\mathcal{A}_1+\mathcal{A}_2+\mathcal{A}_3} \langle i, \alpha \rangle$ if and only if $a_i \vDash_{\mathcal{A}_i} \alpha, \forall i \in \{1, 2, 3\}$.

To make this example concrete, suppose that classifications $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ are given by the classification tables in following Table . Note that each abstract situation (row) is labeled. Then the first six rows classification table for $\mathcal{A}_1 + \mathcal{A}_2 + \mathcal{A}_3$ are shown below.

$\vDash_{A_1}$	α	β	δ	γ
a_1	1	0	1	0
a_2	0	1	1	0
a_3	0	1	0	0

$\vDash_{A_2}$	α	β	ρ
a_4	1	0	0
a_5	1	1	1
a_6	0	1	0

$\vDash_{A_3}$	α	ρ	σ	τ
a_7	0	1	0	1
a_8	0	1	1	0

$\vDash_{A_1+A_2+A_3}$	$\langle 1, \alpha \rangle$	$\langle 1, \beta \rangle$	$\langle 1, \delta \rangle$	$\langle 1, \gamma \rangle$	$\langle 2, \alpha \rangle$	$\langle 2, \beta \rangle$	$\langle 2, \rho \rangle$	$\langle 3, \alpha \rangle$	$\langle 3, \rho \rangle$	$\langle 3, \sigma \rangle$	$\langle 1, \tau \rangle$
(a_1, a_4, a_7)	1	0	1	0	1	0	0	0	1	0	1
(a_1, a_4, a_8)	1	0	1	0	1	0	0	0	1	1	0
(a_1, a_5, a_7)	1	0	1	0	1	1	1	0	1	0	1
(a_1, a_5, a_8)	1	0	1	0	1	1	1	0	1	1	0
(a_1, a_6, a_7)	1	0	1	0	0	1	0	0	1	0	1
(a_1, a_6, a_8)	1	0	1	0	0	1	0	0	1	1	0

4.4 Small Satellite Example

As an example, consider a system of three small satellites, S_1, S_2, and S_3, whose goal is to image the auroral ovals that exist around both magnetic poles of Earth and thereby study the impact of the solar wind on the magnetosphere. S_1 measures the intensity of the auroral brightness; if this is over a given threshold, then S1 measures the magnetic disturbance density and sends a message to S_2 and S_3 to begin imaging. S_2 and S_3 do the same thing from different perspectives, viz., take an image of the auroral oval to determine its extent.

For types for S_1, we distinguish auroral brightness below the threshold (LAB, "Low Auroral Brightness") and at or above the threshold (HAB, "High Auroral Brightness"). We arbitrarily partition the magnetic density measurement, a continuous magnitude, into a small number (viz., three) of ranges for simplicity, giving types LMD ("Low Magnetic Density"), MMD ("Medium"), and HMD ("high"). For S_2, we arbitrarily partition the auroral extent, a continuous magnitude, into three ranges: LAE_2 ("Low Auroral Extent"), MAE_2 ("Medium"), and HAE_2 ("High"). We recognize the same types for S_3 but use subscript "3" in place of "2": LAE_3, MAE_3, and HAE_3.

The classification table for S_1 is shown below. Some of the constraints derived from this table are the following, where (1) indicates that, if the auroral brightness is high, then there is a magnetic density measurement of high, medium, or low, and (2) indicates that the auroral brightness is (unconditionally) high or low.

1. $HAB \vdash LMD, MMD, HMD$
2. $\emptyset \vdash \{HAB, LAB\}$

The classification table for S_2 is shown below. The obvious constraints here are that there must be a measured auroral extent, low, medium, or high ($\emptyset \vdash \{LAE_2, MAE_2, HAE_2\}$), and it cannot be, for example, low and medium at the same time ($\{LAE_2, MAE_2\} \vdash \emptyset$). The classification table for S_3 is identical to that for S_2 but for the subscripts.

$SAT1$	HAB	LAB	LMD	MMD	HMD
1_a	0	1	0	0	0
1_b	1	0	1	0	0
1_c	1	0	0	1	0
1_d	1	0	0	0	1

$SAT2$	LAE_2	MAE_2	HAE_2
2_a	1	0	0
2_b	0	1	0
2_c	0	0	1

The classification table for the core will be analogous to the core classification produced in the previous example but will induce some constraints that cross system parts. At this point, it is convenient to enlarge our system to include a classification for the "part" that is observed by the satellites, namely, the magnetosphere and solar wind as well as the auroral oval produced by their interaction. The real situations classified with the types we have picked out so far certainly include these aspects of the environment and various types that characterize them. We consider here only four types that relate to this environmental part. Let AL indicate that the magnetosphere and solar wind are aligned (within some

level of tolerance), and let NAL indicate that they are not aligned. Let $COND$ indicate that this part is in a state where ionospheric conductivity could be exceptional, and let $NCOND$ indicate the denial of this. If we were to construct a classification table relating these types to types for the other three "parts" of the system (i.e., the satellite classifications), we could use our techniques to derive the following (and many more) constraints.

1. $COND \vdash AL$
2. $\{MMD, HAE_2, HAE_3\} \vdash AL$
3. $\{HMD, HAE_2, MAE_3\} \vdash COND$

Constraint (1) indicates that, if the situation is such that ionospheric conductivity could be exceptional, then the magnetosphere and solar wind are aligned. Constraint (2) indicates that, if the magnetic density is medium and the auroral extent is high from the perspective of both S_2 and S_3, then the magnetosphere and solar wind are aligned. And (3) indicates that, if the magnetic density is high and the auroral extent is high from the perspective of S_2 but medium from the perspective of S_3, then ionospheric conductivity could be exceptional.

5 Maintaining and Communicating Information on Real Situations

Consider now the *situation table* with the information an IKBS has on various real situations. We contrast the situation table with the classification table of a satellite, which is where design starts. The classification table has a column for each type relevant to the satellites sensing capability and the behavior of the monitored region. The rows indicate the combinations of types (columns) that may occur together in an observed situation. That is, the rows specify all the realizable abstract situations by indicating the types realized in them.

In contrast, the situation table has a row for each real situation observed. Like a classification table, a situation table has a column for each observable type, but the situation table needs information on more types. An IKBS has information on a situation not only by observation but also by virtue of utterance situations and by inferring information from information it already has. The descriptive content of an utterance may involve types observable by any satellite in the system, so the situation table must have columns for all these types. The IKBS has its own theory for inferring new information, but, since the IKBS is part of a distributed system, the theory of the whole, or core, is relevant as well for filling out entries in the situation table. Since the "whole" or core does not correspond to a physical entity over and above the "parts" (satellites), inferencing done with the theory of the core must be delegated to these parts. The easiest approach is to assume that all IKBSs have, besides their own theories, the theory of the core.

The general picture, then, has an IKBS, on observing a real situation, make an entry for it in its situation table, indicating what types were observed. When the descriptive content of an utterance relates to this situation, its table entry

may be filled by checking further types. It is also possible that observation may fill in an entry over an extended period. As information on the situation becomes available, various sequents of the local or core theory may allow further information to be inferred and recorded in the entry for the situation. A variation of this picture has the IKBS first finding out about the real situation in question via another's utterance. Note that issues we avoid in this paper include how real situations are denoted, how they are related as parts and wholes, and to what extent something true in a part is also true of the whole. These issues are addressed in the standard literature cited, and the part-whole relation has been extensively studied in terms of lattice structures [25].

The conditions for filling in an entry of the situation table because of a relevant utterance needs to be addressed in more detail since the IKBS may fail to interpret an utterance or the content of what is uttered may be inconsistent with the information the IKBS has. The descriptive content of what may be considered a normal utterance is what we call an *utterance proposition*, of the form $a : \Delta$, where a is a token and Δ is a set of types and type placeholders. For each type, we assume there is a corresponding placeholder $?_\alpha$. If $\alpha \in \Delta$, then part of what the utterance asserts is $a \vDash_{\mathcal{A}} \alpha$, while if neither $\alpha \in \Delta$ nor $?_\alpha \in \Delta$, part of what it asserts is a $a \nvDash_{\mathcal{A}} \alpha$; we cannot have both $\alpha \in \Delta$ and $?_\alpha \in \Delta$. If $?_\alpha \in \Delta$, then the utterance says nothing about a being of type α. If the content of an utterance is consistent with an IKBS's theory (as discussed in Section IV.B), then it is included in the IKBS's situation table unless it is inconsistent with an entry already in that table. The situation table also includes results of observations, and it contains at most one entry for a given real situation (i.e., token). (As an IKBS's classification table constitutes in large part its semantic memory, its situation table constitutes its episodic memory.) We say that an utterance proposition $a : \Delta$ is in the situation table if the situation table has an entry for situation a and that entry records types and type placeholders as per Δ. If the situation table includes the utterance proposition $a : \Delta_1$ and the IKBS perceives an utterance whose descriptive content is $a : \Delta_2$ with the same token a, then the utterance proposition for a in the situation table can be updated with the information $a : \Delta_2$ as long as, for all types α of the IKBS classification, α is not in one of Δ_1 or Δ_2 but not the other. Violation of this condition indicates that the two utterance propositions are inconsistent. It may be the case, though, that α is in one but $?_\alpha$ is in the other, or α is not in one but $?_\alpha$ is in the other. Then the situation table entry for a, $a : \Delta_1$, is updated by replacing any $?_\alpha \in \Delta_1$ with α if $\alpha \in \Delta_2$ and by removing any $?_\alpha \in \Delta_1$ if $\alpha \notin \Delta_2$. If an IKBS succeeds in incorporating an utterance proposition into its situation table, we say that it has *interpreted* that utterance proposition.

In addition to the utterances just addressed, whose descriptive contents are utterance propositions, we allow utterances whose contents are *partial* utterance propositions; we call such utterances *p-utterances* and their descriptive content *p-propositions*. A p-proposition is again of the form $a : \Delta$, but now, for any type α, if $\alpha \notin \Delta$, then $a \nvDash_{\mathcal{A}} \alpha$ is not being asserted; rather, the p-proposition has nothing to say about a being of type α. A p-utterance presents an occasion for

an IKBS to infer (using its theory) additional types for tokens, as illustrated by the paradigmatic case of inferring *fire* from *smoke*. A p-utterance could be used as a query, when one IKBS utters a p-utterance, and another utters a (normal) utterance in reply, providing missing types if they in fact are uniquely determined by the theory. A p-proposition $a : \Delta_1$ is inconsistent with an utterance proposition $a : \Delta_2$ in the situation table if there is at least one type α such that $\alpha \in \Delta_1$ but $\alpha \notin \Delta_2$. If the p-proposition is not inconsistent with any utterance proposition in the IKBS's situation table, then we say that the IKBS *interprets* that p-proposition; this is the case whether or not it is able to uniquely determine additional types for the token.

There are several ways an IKBS may fail to interpret the descriptive content of an utterance proposition or a p-proposition and hence fail to interpret the utterance or p-utterance itself. We say that an IKBS is *acquainted* with a token a if there is an utterance proposition $a : \Delta$ in its situation table, and we say that it is *acquainted* with a type α if α is among the types labeling the columns of its situation table. If an IKBS is unacquainted with a type, it is normally because that type is new and not yet incorporated into the IKBS's computational structures. An IKBS cannot interpret an utterance proposition or p-proposition $a : \Delta$ if there is a type $\alpha \in \Delta$ with which it is not acquainted, and it cannot interpret a p-proposition $a : \Delta$ if it is not acquainted with token a. Failure of interpretation due to lack of acquaintance amounts to a failure to understand. The other way interpretation can fail arises from inconsistency with content of the situation table, as described above.

6 Conclusion

As part of our engineering methodology for small satellite systems that is reliable, formal and results in a "correct-by-construction" design, we presented in this paper a knowledge-based design framework for ensuring that (1) each satellite has a consistent theory it can use to infer new information from information it perceives and (2) the theory for the entire system is consistent so that a satellite can infer new information from information communicated to it, and it can assess purported information from fellow satellites. The point of departure for our framework, from the previous RFD process, is Barwise's channel theory, founded on category theory, and allied work on situation semantics and situation theory. Each small satellite is viewed as an intelligent knowledge base system (IKBS) consisting of classifications in the sense of channel theory. A classification consists of tokens (e.g., observed situations) and types (e.g., features of a situation, such as a certain kind of event) as well as a binary relation that classifies tokens with types. Each IKBS has a semantic part, namely, the classifications with which it is endowed, and a syntactic part, which is a logic or theory of classification that allows each satellite to infer new information from observed and communicated situations. Communication (as per situation semantics) is viewed in terms of utterances; the descriptive content of an utterance is an assertion that a given token is of a given set of types. The core of a system of

classifications, as explained in this paper, is a category-theoretic construct that amalgamates the several classifications; the core and the individual classifications essentially form the "whole" and the "parts" of what is termed a channel. We show how to derive the theory for an IKBS and for the system core, and we show how to check whether a given requirement is derivable from or consistent with a theory.

References

1. Edmonson, W., Herencia-Zapana, H., Neogi, N., Moore, W., Ferguson, S.: Highly confident reduced life-cycle design process for small satellite systems: Methodology and theory. In: Complex Systems and Data Management Conference. (2012)
2. Edmonson, W., Chenou, J., Neogi, N., Herencia-Zapana, H.: Small satellite systems design methodology: A formal and agile design process. In: IEEE International Systems Conference. (2014)
3. Barwise, J., Seligman, J.: Information Flow The logic of Distributed Systems. Cambridge Tracks in Theoretical Computer Science. (1997)
4. Barwise, J., Perry, J.: Situations and Attitudes. MIT Press (1983)
5. Devlin, K.: Logic and Information. Cambridge University Press (1991)
6. MacLane, S.: Categories for the Working Mathematician. Springer-Verlag (1998)
7. Simmons, H.: An Introduction to Category Theory. Cambridge University Press (2011)
8. Awodey, S.: Category Theory. Oxford University Press (2010)
9. Lawvere, W.F., Schanuel, S.H.: Conceptual Mathematics A first Introduction to Categories. Bufflo Worshop Press. (1991)
10. Pierce, B.C.: Basic Category Theory for Computers Scientists. MIT Press. (1991)
11. Barr, M., Wells, C.: Category Theory for Computing Science. Prentice Hall (1998)
12. Fiadeiro, J.L.: Categories for Software Engineering. Springer (2005)
13. Schorlemmer, M., Kalfoglou, Y., Atencia, M.: A formal foundation for ontology-alignment interaction models. International Journal on Smantic Web and Information Systems (2007)
14. Kalfoglou, Y., Schorlemmer, M.: Formal support for representing and automating semantic interoperability. In: The Semantic Web, Springer, Heidelberg (2004)
15. Kent, R.E.: Semantic integration in the information flow framework. semantic interoperability and integration. In: Semantic Interoperability and Integration, Dagstuhl Seminar Proceedings, Wadern: Dagstuhl seminar Proceedings. (2005)
16. Spivak, D.I.: Functional data migration. Information and Computation (2012)
17. Spivak, D.I., Kent, R.E.: Ologs: A categorical framework for knowledge representation. PLoS ONE 7 (1) (2012)
18. Diskin, Z., Maibaum, T.: Category theory and model-driven engineering: From formal semantics to designs patterns and beyond. In: Workshop on Applied and Computational Category Theory
19. Goguen, J., Burstall, R.: Institutions: Abstract model theory for specification and programming. Journal of Association for Computing Machinery (1992)
20. Goguen, J.: Information integration in institutions. In: Moss, L. (Ed.) Memorial Volume for Jon Barwise. Indiana University Press, Bloomington. (2004)
21. Bazhanov, K., Obiedkov, S.: Optimizations in computing the duquenneguigues basis of implications. Annals of Mathematics and Artificial Intelligence (2014)
22. Armstrong, W.W.: Dependency structures of data base relationships. (1974)

23. Beeri, C., Dowd, M., Fagin, R., Statman, R.: On the structure of armstrong relations for functional dependencies. Journal of the ACM (1984)
24. Sam Owre, J.M.R., Shankar, N.: pvs: A prototype verification system. In: 11th International Conference on Automated Deduction (CADE). (1992)
25. Link, G.: Algebraic Semantics in Language and Philosophy. CSLI Lecture Notes No. 74 (1998)

Author Index

M

N

O

P

R

S

W