A Semantic Dashboard Description Language for a Process-oriented Dashboard Design Methodology

Maximilien Kintz
Fraunhofer IAO
Nobelstraße 12
70569 Stuttgart Germany
+49 711 970-2182
maximilien.kintz@iao.fraunhofer.de

ABSTRACT
Monitoring and controlling business processes is a challenging task: different data sources need to be combined, appropriate visualizations need to be defined to observe certain goals or Key Performance Indicators (KPI). In this paper, a new semantic dashboard description language used in a process-oriented dashboard design methodology is introduced, to help users focus more on business processes and actual goals and less on technical aspects of monitoring and controlling infrastructure. A reference implementation is described and plans for future improvements of the methodology are introduced.

Keywords
Dashboards, business processes, monitoring, controlling, XML

INTRODUCTION
The monitoring and controlling of business processes places users in front of difficult challenges: multiple and sometimes incompatible data sources have to be integrated, specific needs for the monitoring of precise goals or controlling possibilities of certain process definition values require complex and hard to use Business Intelligence (BI) solutions usually targeted at experts in information analysis, not in the particular business domain being monitored.

As the importance of BI continuously increases (as stated in [9]), to help solve these difficulties, we propose a new process-oriented dashboard design methodology, part of a larger process monitoring methodology and relying on a semantic XML-based dashboard description language.

The remained of this paper is organized as follows: first, the state of the art in dashboard design and usage, related work in process and model driven design methodologies, as well as description languages for semantic user interfaces are presented. Then, the new process-oriented monitoring and dashboard design methodologies are presented. The dashboard description language and its semantic characteristics are then described in detail. A prototype reference implementation is introduced and first results are assessed. Finally, future work is presented with conclusions in the last section.

STATE OF THE ART AND RELATED WORK
Dashboards are nowadays widely used for monitoring and analysis of business processes. Numerous companies such as IBM [14], SAP [15], Tableau Software [16] or TIBCO Spotfire [17], to name a few well-known vendors, offer complete BI or information visualization solutions.

Rules and best practices for the design and use of dashboards are also already widely investigated: from a conception and use point of view, Eckerson proposes in [11] detailed advice. From an information visualization point of view, Few, in [12], gives precise and clear guidelines to follow in order to create dashboards that are easy and efficient to use.

A model-driven design method using BPMN process models to derive dashboards has also already been proposed in [7]. However, this method required information visualization knowledge from its users in order to create efficient designs. The method presented in this paper tries to include this knowledge in the "intelligence" of the dashboard generation and to automate as much as possible. Furthermore, the generation of a full monitoring infrastructure was not foreseen, and possibilities of process controlling supported by our approach were not investigated.

To efficiently describe the dashboards generated by the methodology presented in this paper, a new dashboard description language is introduced. This language helps describing files containing information on graphical visualization, datasets, interaction and controlling elements, alerting and process semantic. Already existing languages offering similar but more limited capabilities (as the semantic and controlling aspects are often forgotten and a higher importance is given to styling and layout) include Vizql [18] derived from the Polaris [23] framework, nowadays integrated in a different form in the Tableau Software solutions and unfortunately proprietary and closed. Standardized serializations for charts in XML

already exist, such as ChartsML [22] used in a Firefox [25] browser extension or in the chart design tool suite FusionCharts [24]. Other graphical user interface description languages such as Adobe Flex [19] are related. Flex focuses on interactions and allows using XML to design charts but is limited as data sources are concerned. Scalable Vector Graphics (SVG) [20] is mighty but too generic for the use case described in this paper. All these languages tend to be very generic and thus too complex to be easily managed by non-specialists users, and do not offer a sufficient level of semantic to be fully appropriate for our process monitoring and controlling use-case.

PROCESS-ORIENTED SYSTEM MONITORING AND CONTROLLING

The process-oriented dashboard design methodology presented in this article extends a full business process-model-driven monitoring methodology and infrastructure currently in development but already largely implemented and described in [3] and [4] and summarized on Figure 1.

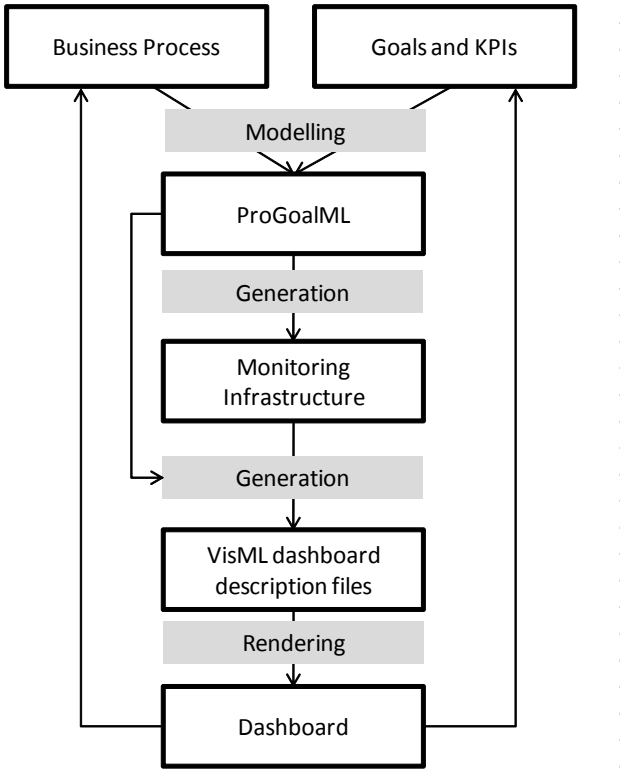

Figure 1: Overview of the process monitoring methodology

Overview of the a.pro Process Monitoring Methodology

The first step of the method involves the users of a system to be monitored defining goals and KPIs, and modeling the business process they want to observe using the BPMN [6] notation. In the process model, possible control points for further process controlling are also indicated. This information, including which goals and KPIs are of interest to which users (i.e. a role concept) is described and stored in an XML file in the specifically created format *ProGoalML*. As this modeling and description operation can be complex, it is performed by an expert, using user input.

Based on this XML file, a whole monitoring infrastructure is then automatically set-up, and code stubs to be integrated in the running process engine in order to send data to a real-time monitoring service using a CEP engine (using the methodology described in [8]) and to a data warehouse for archiving are generated, to be integrated by the responsible IT department in the system.

When this infrastructure has been set up, the dashboard generation phase can begin.

A business-process-model-driven dashboard design methodology

Taking as input the ProGoalML file containing information on the process model, roles and KPIs, as well as access to the CEP engine for real-time information, to the data warehouse for historical information and to control stubs web services if controlling of the process is wanted and allowed by the process model, the VisML generation is started.

The first VisML file generation process includes the following steps:

1. Matching goals to data types (as is already done in the process monitoring methodology for the creation of the CEP engine and data warehouse);
2. Matching data types to visualizations (cf. Figure 2), answering typical questions and following processes in a way similar to that defined in [21] and using best practices for the choice and styling of charts described in information visualization literature, for example in [1]. To allow for a flexible and easy update of the matching algorithm, the matching configuration is stored in an XML file containing pairs of ProGoalML goals descriptors and of VisML charts descriptors;
3. Matching visualizations to the correct data source (CEP engine Web service interface for real time data, data warehouse for historical data) and defining appropriate options for drill-down and other interactions;
4. Defining proper alerting and controlling commands, using goals defined in the ProGoalML file and information from the process model;
5. Generating a basic dashboard layout for each role, by enabling only the relevant visualizations for a specific role, and sorting them according to both priorities defined in the roles and views model accompanying the ProGoalML and to semantic criteria (i.e. placing visualizations related to the same process steps next to each other).

As usual with model-driven graphical interface generation [5] (dashboards being here a specific type of user interface),

the mapping described in steps 1 to 3 is of particular importance.

Once one or more (depending on the number of roles and views specified) VisML files have been generated, these can be loaded in a compatible dashboarding engine. There, the users can use and modify them (by hiding visualization, switching places or changing sizes, disabling some alerts, etc.) as necessary, and thus recreate customized VisML files, should they not be entirely satisfied with the automatically generated ones.

Data type	*Visualization*
Composition, categories	Bar chart
Comparison over time, distribution	Line chart
Single value	Number, possibly sparkline
Difference actual value vs. objective	Bullet graph

Figure 2: Mapping data type to visualization

Although the customization is possible at any time, a normal use case would consist on a limited number of iterations for the modifications of the VisML files, after which monitoring and controlling tasks would simply be performed as in traditional BI and monitoring solutions.

The components of the process-model-driven dashboard design methodology and their interactions are presented on Figure 3.

A SEMANTIC DASHBOARD DESCRIPTION LANGUAGE

The dashboards created with the new dashboard design method have specificities such as semantic information and controlling possibilities which make it necessary to describe them with a new language, different than the ones currently used by dashboard solutions vendors. We call this new language *VisML* for *visualization markup language*, as it could theoretically be used to describe any kind of visualization, not only dashboards.

Design goals for the VisML language were to be highly semantic (focusing on describing the meaning of elements and leaving technical aspects such as layout or styling to the dashboard rendering engine, thus making it possible to adapt the same VisML dashboard to a desktop, tablet or smartphone layout, for example), easily human readable, and overall as simple as possible (i.e. limited to the information that is absolutely necessary to render the dashboard).

The overall structure of a dashboard file described in the new semantic dashboard description language is presented in the next paragraphs, an example is shown on Figure 4.

Overview

A dashboard file should contain the information needed by a software tool to render a graphic view of the dashboard as specified by the user, and allow for required interactions such as drill-down or controlling. To achieve this, it appeared necessary to include generic information on the dashboard, an exhaustive list of all visualizations that compose the dashboard, some specific conditions for monitoring use cases such as alerting information, and references to the data to be displayed.

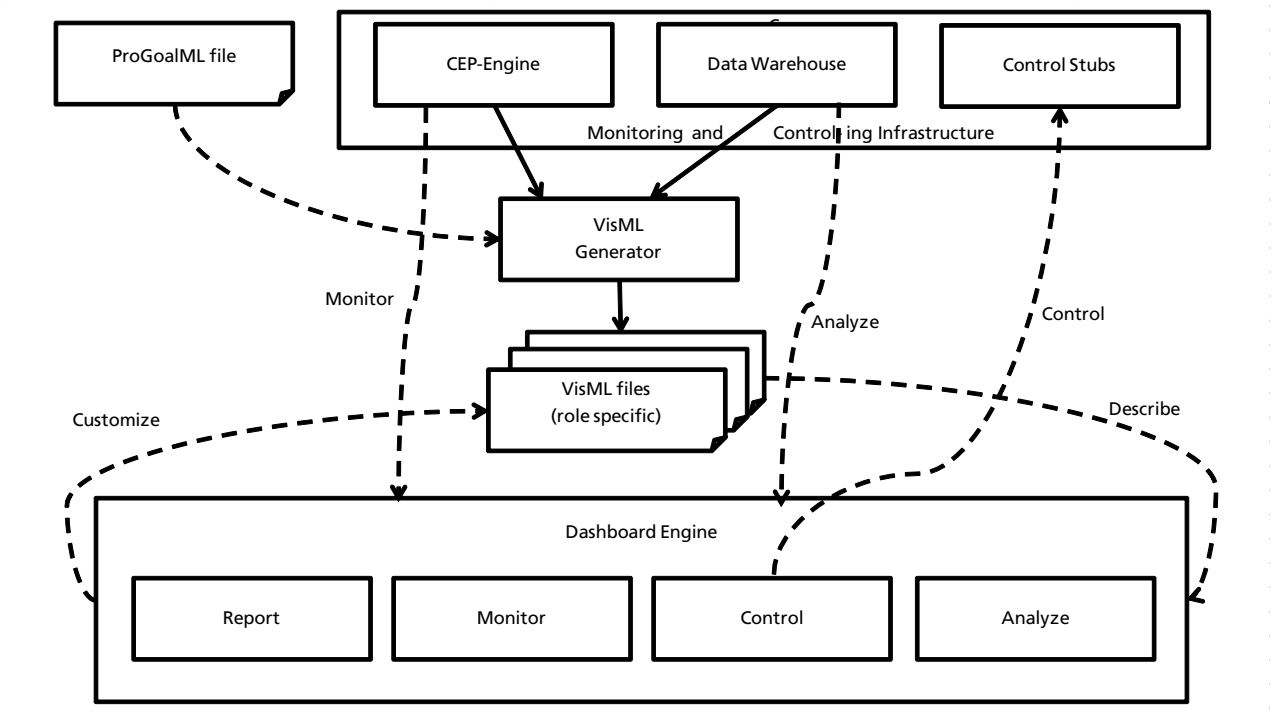

Figure 3: The components of the dashboard design methodology

We use XML [10] to create documents that are easily human and machine readable.

The structure of a dashboard file is therefore specified as an XML file containing a top element *visml*, itself containing exactly four sub-elements:

1. The element *meta*, containing generic information such as title, author, etc.
2. The element *dashboard*, containing a collection of *visualization* elements, each describing a specific visual part of the dashboard.
3. The element *alerting*, containing a series of alert elements, specifying conditions to be monitored and actions to be triggered when a condition is reached.
4. The element *data*, containing a series of data sources and sets.

In the following paragraphs, we describe the role and structure of each of these four sub-elements.

The Element META

The element *meta* (for metadata) contains generic information describing the dashboard. The element is composed of two sub-elements: *description* and *semantic*. The sub-element *description* is composed of a mandatory element *title*, indicating a title given to the dashboard (used to be display as a window title, for storage, etc.), zero or more *author* elements, used to store the names of the dashboard authors, and two optional *datetime* and *comment* elements, used to store a modification date and time of the dashboard structure (for versioning) and additional comments.

The element semantic is described in a specific section.

The Elements DASHBOARD and VISUALIZATION

The main element of a dashboard description document is the element *dashboard*. It represents the part of the dashboard that must be visually rendered by the supporting tool and presented to the user.

A dashboard consists in a collection of *visualization* elements. Each visualization represents a graphical display of information, theoretically in any possible form. However, the dashboard description language focuses on the description of business dashboards and therefore privileges those displays that are typical in such dashboards, such as line graphs, bar graphs, tables and numbers with sparklines [2], or bullet graphs [6].

A visualization is defined by its attributes *category* and *type*. The possible *category* values are *chart*, *graph*, *map*, *diagram*, *table*, *text* and *other*. This list is a slightly adapted version of the classification of information graphics introduced in (Harris, 1999). *Text* has been added because simple textual messages or numbers are often useful on dashboards; other ensures the completeness of the specification.

Other characteristics of a visualization are a *dataset* (see related section), a *title* and optional *description*, an element *interaction* that contains a list of possible interaction capabilities such as *zoom*, *pan* or *drill-down*, *style* and *semantic* information.

```
<?xml version="1.0" encoding="UTF-8"?>

<visml version="" xmlns="...">

<meta>
          <title>Title of the dashoard</title>
          <author>Author name</author>
...
</meta>

<dashboard>
<visualization dataset="set1" id="vis1" category="chart"
type="line">
          <title>Line chart title</title>
          <interaction type="click">
          <scope>...</scope>
          </interaction>
          </visualization>
...
</dashboard>

<alerting>
<alert id="1" dataset="set1"
<condition><![CDATA[saving > 10]]></condition>
<mailNotification frequency="EVERY 30minutes">
<email>mail@example.org</email>
<message>This is the e-mail alert message.</message>
</mailNotification>
</alert>
...
</alerting>

<data>

<datasources>
<database id="source1">…</database>
...
</datasources>

<datasets>
<dataset datasource="source1" id="set1" format="..."
type="...">
<semantic><relation id="..." type="..."/></semantic>
<query><![CDATA[SELECT X AS NAME, COUNT(COST) AS VALUE
FROM costs WHERE COST <= 499;]]></query>
</dataset>
...
</datasets>
...
</data>

</visml>
```

Figure 4: Excerpt from an example VisML dashboard description file

The Element ALERTING

It is possible to define alerting conditions and messages in a VisML file. When a certain dataset meets a condition, the specified corresponding alert message can pop-up in the dashboard window or can be sent by e-mail to a list of recipients. Other actions can of course be imagined (sending an SMS or even impacting a system), as long as they can properly be supported by the monitoring and controlling engines.

An *alert* element typically consists of a *condition* referring to a dataset and one or more notifications, each with their own frequency and messages.

The Element SEMANTIC

A specify of the VisML dashboard description language it its ability to not only describe the appearance of

visualizations but also their meaning and links to actual business processes or process steps, which can easily be done since the dashboard design method is business process model-driven.

For each dashboard in the element *description* and for each individual data set, relations to process steps identified by their ID as specified in the business-model part of the ProGoalML file can be specified. The *semantic* simply consists of one or more *relation* elements, each having three attributes: *id* for the process-step being referenced, *priority* for differentiating relations when several are attributed to the same visualization, and *type*.

This semantic information is then used by the dashboard rendering engine to help sort and place the visualizations in meaningful ways, and to better manage links to the actual process steps for use-cases such as controlling of the business process.

The Elements DATA, DATASOURCE and DATASET

The element *data* contains two main sub-elements: *datasources* and *datasets*.

The element *datasources* contains a list of sources of dashboard data. A data source is a reference (for example, the connection parameters) to a specific database, a Web service delivering data, or possibly a document containing data in a format supported by the dashboarding engine (this use-case is not supported in the current implementation).

The element *datasets* contains a series of sets of dashboard data, derived from the data sources specified before. A *dataset* is a specific reference to a data source table, a query on a database or a list of values to be retrieved from a Web service. A dataset can also be defined as a *meta-dataset*, i.e. as the resulting combination of other *dataset* elements already defined in the VisML file. The *dataset* also contains lists of labels to be mapped to value categories and later displayed as labels on the visualization, for example on the axis of graphs. Each dataset may contain several relations to process models or steps defined with the element *semantic*.

Linking VisML with common BI solutions

As the VisML dashboard description files are simple XML documents, they can be processed, for example with XSLT style sheets, to be transferred into dashboard files compatible with other already existing BI solutions. Necessary conditions to that end are that the vendor-specific dashboard documents do not require more information than is available in the VisML file, that the vendor-specific language is documented and the proper interpreter is implemented.

EXAMPLE IMPLEMENTATION FOR INSURANCE CLAIMS MANAGEMENT

Large parts of the a.pro process monitoring methodology and of the dashboard design methods described in this paper have already been implemented. The VisML semantic dashboard description language is currently in productive use in a lightweight dashboard for the monitoring of an insurance claims management tool, easily allowing users to simply edit and adapt their dashboards from a Web browser-based full Javascript interface. An example VisML dashboard is presented on Figure 5.

Feedback gathered from the use of the lightweight VisML dashboards show that the simplicity of the language is an important aspect, as it allows users to immediately adapt the dashboard to their needs if the generated version wasn't satisfying. Although interactions are supported by the

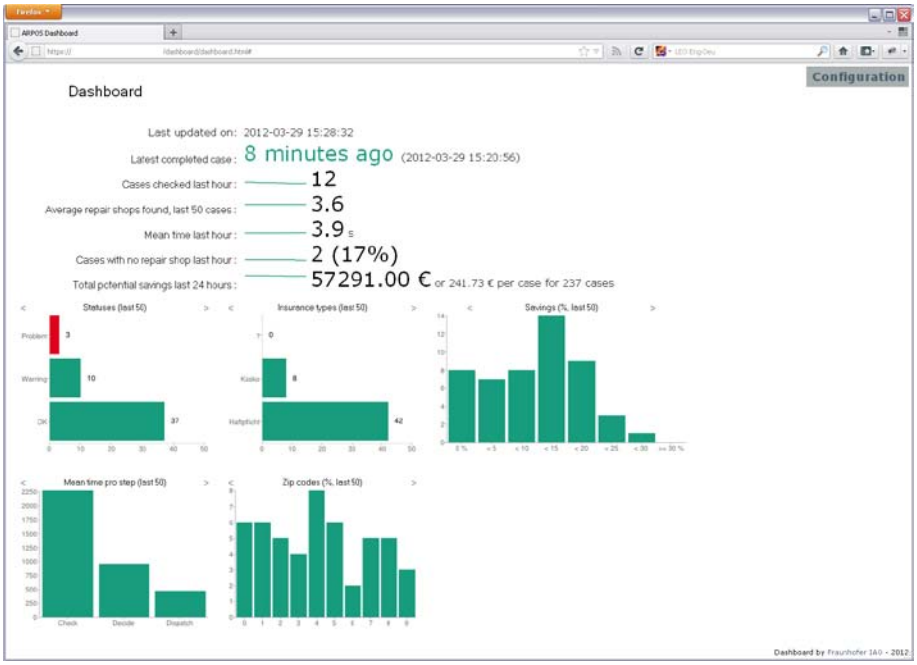

Figure 5: An example VisML dashboard

language specification, they do not for all use-cases need to be supported by the rendering engine, as users of dashboards for monitoring purposes do not necessarily always need to perform advanced exploratory data analysis tasks.

The full automatic generation of the dashboards and the use of control points to actively impact the business process being monitored still needs to be implemented and tested.

CONCLUSION

To help overcome known challenges of business process monitoring, such as the difficulty to built appropriate dashboards from complex data sources to best monitor given goals, a new semantic dashboard description language used in a process-oriented dashboard design methodology was introduced. This methodology is part of a larger business-process monitoring and controlling methodology. Large parts of these solutions have already been implemented and successfully used in production.

The process monitoring solution we presented offers several characteristics of ubiquitous computing as defined in [26]: it allows for the easy and seamless interaction and monitoring of complex and multiple IT systems, it helps the users forget about technical details and focus on business processes and goals, and as the VisML dashboard description language we introduced emphasizes semantics over styling, it can be used to render adapted views of the dashboards on many different systems, such as single or multi-screen desktops, tablets or even smartphones.

Future work focuses on the improved fully automated generation of dashboards and on the possibilities of active business process controlling using the monitoring dashboards as configuration panels.

REFERENCES

1. S. Few. *Effectively Communicating Numbers: Selecting the Best Means and Manner of Display*, 2005. http://www.rit.edu/cla/cpsi/SRResources/Effectively%20Communicating%20Numbers.pdf
2. E. R. Tufte. *Beautiful Evidence. Graphics Press, 2006.*
3. F. Koetter and M. Kochanowski. Goal-Oriented Model-Driven Business Process Monitoring using ProGoalML. In *Proceedings of the 15th International Conference on Business Information Systems (BIS 2012*) (in press), 2012.
4. F. Koetter, A. Weisbecker and T. Renner. Business Process Optimization in Cross-Company Service Networks – Architecture and Maturity Model. In *Proceedings of the 2012 Annual SRII Global Conference*, 2012.
5. T. Schlegel.; M. Raschke.; M. Knittig.; A. Dridiger.; S. Wokusch. and C. Taras. Evaluation of Current User Interface Generator Frameworks for Graphical Interactive Systems. In *Proceedings of the IADIS International Conference Interfaces and Human Computer Interaction 2010*, 2010.
6. S. Few. *Bullet Graph Design Specification*, 2010. http://www.perceptualedge.com/articles/misc/Bullet_Graph_Design_Spec.pdf
7. P. Chowdhary, T. Palpanas, F. Pinel, S.-K. Chen, and F. Y. Wu. Model driven dashboards for business performance reporting. In *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)*, 2006.
8. T. Schlegel; S. Dusch and K. Vidackovic. Interaction- and Event-Based Management of Processes in Service-Oriented Infrastructures. In *Proceedings of the 6th I*PROMS virtual conference*, 2010.
9. J. Hagerty, R. L. Sallam, and J. Richardson. *Magic quadrant for business intelligence platforms*. 2012.
10. W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, 2008.
11. W.W. Eckerson. *Performance Dashboards: Measuring, Monitoring, and Managing Your Business*. John Wiley & Sons, Inc., 2011.
12. S. Few. *Information Dashboard Design*. O'Reily Media, Inc., 2006.
13. OMG. *Business Process Model and Notation (BPMN) Version 2.0*, 2009.
14. IBM Business Analytics. http://www-142.ibm.com/software/products/us/en/category/SWQ00
15. SAP Businessobjects Business Intelligence Solutions. http://www.sap.com/solutions/sapbusinessobjects/large/business-intelligence/index.epx
16. Tableau Software. http://www.tableausoftware.com/
17. TIBCO Spotfire. http://spotfire.tibco.com/
18. P. Hanrahan. Vizql: A language for query, analysis and visualization. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, 2006.
19. Adobe Flex. http://www.adobe.com/products/flex.html
20. W3C. SVG Working Group. http://www.w3.org/Graphics/SVG/
21. X.J. Li; T. Schlegel; M. Rotard and T. Ertl. A Model-Based Graphical User-Interface for Process Control Systems in Manufacturing. In *Proceedings of the Intelligent Production Machines and Systems - 2nd I*PROMS Virtual International Conference*, 2006.
22. T. Saito. *ChartML*. 2008. http://www.onsaito.com/csdv/chartMLindex.xhtml
23. C. Stolte; D. Tang. and P. Hanrahan. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. In *IEEE Transactions on Visualization and Computer Graphics* 8, 2002.
24. FusionCharts. *Charts XML Reference*. http://docs.fusioncharts.com/free/
25. Mozilla Firefox. http://www.mozilla.org/en-US/firefox/new/
26. M. Weiser. Hot Topics: Ubiquitous Computing. In *IEEE Computer*, October 1999.