Preface

Piero A. Bonatti[1], Li Ding[2], Tim Finin[3], and Daniel Olmedilla[4]

[1] Università di Napoli Federico II, Napoli, Italy
bonatti@na.infn.it
[2] Knowledge Systems Lab, Stanford,USA
ding@ksl.stanford.edu
[3] University of Maryland Baltimore County, USA
finin@cs.umbc.edu
[4] L3S Research Center and University of Hannover, Hannover, Germany
olmedilla@L3S.de

Policies are pervasive in web applications. They play crucial roles in enhancing security, privacy and usability of distributed services, and indeed may determine the success (or failure) of a web service. However, users will not be able to benefit from these protection mechanisms unless they understand and are able to personalize policies applied in such contexts. For web services this includes policies for access control, privacy and business rules, among others. There has been extensive research in the area, including the Semantic Web community, but several aspects still exist that prevent policy frameworks from widespread adoption and real world application like for example:

- Adoption of a broad notion of policy, encompassing not only access control policies, but also privacy policies, business rules, quality of service, and others.
- Strong and lightweight evidence: Policies make decisions based on properties of the peers interacting with the system. These properties may be strongly certified by cryptographic techniques, or may be reliable to some intermediate degree with lightweight evidence gathering and validation.
- Policy-driven negotiations may be one of the main ingredients that can be used to make heterogeneous peers effectively interoperate.
- Lightweight knowledge representation and reasoning should also reduce the effort to specialize general frameworks to specific application domains
- Solutions like controlled natural language syntax for policy rules, to be translated by a parser into the internal logical format, will definitively ease the adoption of any policy language.
- Cooperative policy enforcement: A secure cooperative system should (almost) never say no. Whenever prerequisites for accessing a service are not met, web applications should explain what is missing and help the user in obtaining the required permissions.
- Advanced explanation mechanisms are necessary to help users in understanding policy decisions and obtaining the permission to access a desired service.

This volume[5] contains the papers presented at the 2nd International Semantic Web Policy Workshop (SWPW'06) held on Athens in Georgia, USA on November 5th, 2006, in conjunction with the 5th International Semantic Web Conference (ISWC).

In response to the call for papers there were a total number of 15 submissions. These papers were evaluated on the basis of their originality and contribution, technical quality and presentation. Each submission was reviewed by at least 3 programme committee members. The committee finally decided to accept 4 full papers and 6 short/position papers. The program also includes an invited talk by Grit Denker on *Policy Specification and Enforcement For Spectrum-Agile Radios*.

We would like to thank all people who contributed to the success of the workshop including the programme committee members and external referees, who provided timely and indepth reviews of the submitted papers, all authors who submitted papers and all the attendees.

Piero A. Bonatti, Li Ding, Tim Finin, and Daniel Olmedilla
Programme Committee Chairs
SWPW 2006

[5] The management of the workshop as well as the generation of the workshop proceedings greatly benefitted from the EasyChair conference management system.

Workshop Organization

Programme Chairs

Piero A. Bonatti, University of Naples
Li Ding, Knowledge Systems Lab, Stanford
Tim Finin, University of Maryland Baltimore County
Daniel Olmedilla, L3S Research Center & Hannover University

Programme Committee

Anne Anderson, Sun Microsystems
Anupam Joshi, University of Maryland, Baltimore County
Chris Bizer, FU Berlin
Piero Bonatti, University of Naples
Jeffrey M. Bradshaw, Florida IHMC
Li Ding, Knowledge Systems Lab, Stanford University
Naranker Dulay, Imperial College
Tim Finin, University of Maryland, Baltimore County
Lalana Kagal, MIT
Jiangtao Li, Purdue University
Brian LaMacchia, Microsoft
Fabio Martinelli, National Research Council - C.N.R.
Rebecca Montanari, University of Bologna
Wolfgang Nejdl, L3S and University of Hannover
Daniel Olmedilla, L3S and University of Hannover
Norman Sadeh, Carnegie Mellon University
Pierangela Samarati, University of Milano
Kent Seamons, Brigham Young University
William Winsborough, University of Texas at San Antonio

Additional Referees

Sabrina De Capitani di Vimercati
Jinghai Rao
Alessandra Toninelli

Table of Contents

Policy Specification and Enforcement For Spectrum-Agile Radios

Grit Denker

SRI International, 333 Ravenswood Ave, Menlo Park, California 94025
Grit.Denker@sri.com

Because of the centralized, static nature of current spectrum allotment policy, wireless communication is confronting two significant problems: spectrum scarcity and deployment delays. Current systems are significantly impacted by the lack of access to unused spectrum, and existing spectrum management procedures are too inflexible to react to dynamic operational needs. The policies are static and policy changes are very labor intensive.

Existing policies assume 100% spectrum use; however, studies have shown that most assigned spectrum is unused most of the time. This motivated the goals of DARPA's neXt Generation (XG) Communications Program, which envisions opportunistic spectrum access. XG provides technologies for automatic, dynamic, and opportunistic access to unused spectrum. This requires that radios sense and opportunistically adapt to local RF environments and application needs.

In addition, radios must act according to regulatory rules. Spectrum use policies are authored in more than 200 countries and are verified by each host nation. Policies are customizable to location, user, time, frequency and many other parameters, and they change over time. The large number of operating dimensions to be considered makes it difficult to find a solution for the optimal use of radio spectrum. In XG this problem is solved by using platform-independent *Policy Controls* to regulate spectrum access in changing regulatory environments.

We developed an expressive and extensible policy language for describing policies that meet the needs of a wide variety of spectrum regulation bodies, including reuse of ontological concepts, and for supporting efficient reasoning. We also implemented a policy-conformance algorithm that checks the compliance of candidate transmissions using efficient, state-of-the-art reasoning technology. The design of the algorithm is independent of, yet easy to integrate into, any radio design, to encourage competitive, best-of-breed XG radio development.

We will report on our experience in designing the Cognitive Radio Language (CoRaL). CoRaL is a declarative language based on a typed version of classical first-order logic. We will also summarize the results of our evaluation of various logical formalisms with respect to their appropriateness for the XG domain. Finally, we will illustrate the main concepts of the XG Policy Reasoner.

Acknowledgments.

This research was supported by DARPA's neXt Generation (XG) Communications Program under Contract Number FA8750-05-C-0230. (Approved for Public Release, Distribution Unlimited)

Verification, Validation, Integrity of Rule Based Policies and Contracts in the Semantic Web

Adrian Paschke

Internet-based Information Systems, Dept. of Informatics, TU Munich, Germany
Adrian.Paschke@gmx.de

Abstract. Rule-based policy and contract systems have rarely been studied in terms of their software engineering properties. This is a serious omission, because in rule-based policy or contract representation languages rules are being used as a declarative programming language to formalize real-world decision logic and create IS production systems upon. This paper adopts a successful SE methodology, namely test driven development, and discusses how it can be adapted to verification, validation and integrity testing (V&V&I) of policy and contract specifications. Since, the test-driven approach focuses on the behavioral aspects and the drawn conclusions instead of the structure of the rule base and the causes of faults, it is independent of the complexity of the rules and the system under test and thus much easier to use and understand for the rule engineer and the user.

Key words: Declarative Verification, Validation and Integrity (V&V&I), Rule-based Policies / SLAs, Rule Interchange, Test Cases, Test Coverage

1 Test-driven V&V for Rule-based Policies and Contracts

Increasing interest in industry and academia in higher-level policy and contract languages has led to much recent development. Different representation approaches have been propose, reaching from general syntactic XML markup languages such as WS-Policy, WS-Agreement or WSLA to semantically-rich (ontology based) policy representation languages such as Rei, KAoS or Ponder and highly expressive rule based contract languages such as the RBSLA language [2] or the SweetRules approach. In this paper we adopt the rule-based view on expressive high-level policy and contract languages for representing e.g. SLAs, business policies and other contractual, business-oriented decision logic. In particular, we focus on logic programming techniques. Logic programming has been one of the most successful representatives of declarative programming. It is based on solid and well-understood theoretical concepts and has been proven to be very useful for rapid prototyping and describing problems on a high abstraction level. The domain of contractual agreements, high-level policies and business rules' decision logic appears to be highly suitable to logic programming. For instance, IT service providers need to manage and possibly interchange large amounts of SLAs / policies / business rules which describe behavioral, contractual or business logic using different rule types to describe e.g. complex conditional decision logic (derivation rules), reactive or even proactive behavior (ECA rules), normative statements and legal rules (deontic rules), integrity definitions (integrity constraints) or defaults, rule preferences and exceptions (non-monotonic defeasible rules). Such rule types have been shown to be adequately represented and formalized as logic programs (LPs) - see the ContractLog KR [1] developed in the RBSLA project [3]. However, the domain imposes some specific needs on the engineering and life-cycle management of formalized policy / contract specifications: The policy rules must be necessarily modelled evolutionary, in a close collaboration between domain experts, rule engineers and practitioners and the statements are not of static nature and need to be continuously adapted to changing needs. The future growth of policies or contract specifications, where rules are often managed in a distributed way and are interchanged between domain boundaries, will be seriously obstructed if developers and providers do not firmly face the problem of quality, predictability, reliability and usability also w.r.t. understandability of the results produced by their rule-based policy/contract systems and programs. Furthermore, the derived conclusions and results need to be highly reliable and traceable to count even in the legal sense. This amounts for verification, validation and integrity testing (V&V&I) techniques, which are much simpler than the rule based specifications itself, but nevertheless adequate (expressive enough) to approximate their intended semantics, determine the reliability of the produced results, ensure the correct execution in a target inference environment and safeguard the life cycle of possibly distributed and unitized rules in rule-based policy projects which are likely to change frequently.

Different approaches and methodologies to V&V of rule-based systems have been proposed in the literature such as model checking, code inspection or structural debugging. Simple operational debugging approaches which instrument the policy/contract rules and explore its execution trace place a huge cognitive load on the user, who needs to analyze each step of the conclusion process and needs to understand the structure of the rule system under test. On the other hand, typical heavy-weight V&V methodologies in Software Engineering (SE) such as waterfall-based approaches are often not suitable for rule-based systems, because they induce high costs of change and do not facilitate evolutionary modelling of rule-based policies with collaborations of different roles such as domain experts, system developers and knowledge engineers. Moreover, they can not check the dynamic behaviors and the interaction between dynamically updated and interchanged policies/contracts and target execution environments at runtime. Model-checking techniques and methods based e.g. on algebraic-, graph- or Petri-net-based interpretations are computationally very costly, inapplicable for expressive policy/contract rule languages and presuppose a deep understanding of both domains, i.e. of the the testing language / models and of of the rule language and the rule inferences. Although test-driven Extreme Programming (XP) techniques and similar approaches to agile SE have been very successful in recent years and are widely used among mainstream software developers, its values, principles and practices have not been transferred into the rule-based policy and contract representation community yet. In this paper, we adapt a successful methodology of XP, namely test cases (TCs), to verify and validate correctness, reliability and adequacy of rule-based policy and contract specifications. It is well understood in the SE community that test-driven development improves the quality and predictability of software releases and we argue that TCs and integrity constraints (ICs) also have a huge potential to be a successful tool for declarative V&V of rule-based policy and contract systems. TCs in combination with other SE methodologies such as *test coverage measurement* which is used to quantify the completeness of TCs as a part of the feedback loop in the development process and *rule base refinements* (a.k.a. *refactorings*) [17] which optimize the existing rule code, e.g. remove inconsistencies, redundancy or missing knowledge without breaking its functionality, qualify for typically frequently changing requirements and models of rule-based policies and contracts (e.g. SLAs). Due to their inherent simplicity TCs, which provide an abstracted black-box view on the rules, better support different roles which are involved during the engineering process and give policy engineers an expressive but nevertheless easy to use testing language. In open distributed environment TCs can be used to ensure correct execution of interchanged specifications in target execution environments by validating the interchanged rules with the attached TCs.

The further paper is structured as follows: In section 2 we review basics in V&V research. In section 3 we define syntax and semantics of TCs and ICs for LP based policy/contract specifications. In section 4 we introduce a declarative test coverage measure which draws on inductive logic programming techniques. In section 5 we discuss TCs for V&V of rule engines and rule interchange. In section 6 we describe our reference implementation in the ContractLog KR and integrate our approach into an existing SE test framework (JUnit) and a rule markup language (RuleML). In section 7 we discuss related work and conclude this paper with a discussion of the test-drive V&V&I approach for rule-based policies and contracts.

2 Basics in Rule-based V&V Research

V&V of rule-based policy/contract specifications is vital to assure that the LP used to formalize the policy/contract rules performs the tasks which it was designed for. Accordingly, the term V&V is used as a rough synonym for "evaluation and testing". Both processes guarantee that the LP provides the intended answer, but also imply other goals such as to assure the security or maintenance and service of the rule-based system. There are many definitions of V&V in the SE literature. In the context of V&V of rule-based policies/contracts we use the following:

1. *Verification* ensures the logical correctness of a LP. Akin to traditional SE a distinction between structurally flawed or logically flawed rule bases can be made with structural checks for redundancy or relevance and semantic checks for consistency, soundness and completeness.
2. As discussed by Gonzales [4] *validation* is concerned with the correctness of a rule-based system in a particular environment/situation and domain.

During runtime certain parts of the rule based decision logic should be static and not subjected to changes or it must be assured that updates do not change this part of the intended behavior of the policy/contract. A common way to represent such constraints are ICs. Roughly, if validation is interpreted as: "*Are we building the right product?*" and verification as: "*Are we building the product right?*" then integrity might be loosely defined as: "*Are we keeping the product right*?", leading to the new pattern: **V&V&I**. Hence, ICs are a way to formulate consistency (or inconsistency) criteria of a dynamically updated knowledge base (KB). Another distinction which can be made is between errors and anomalies:

- *Errors* represent problems which directly effect the execution of rules. The simplest source of errors are typographical mistakes which can be solved by a verifying parser. More complex problems arise in case of large rule bases incorporating several people during design and maintenance and in case of the dynamic alteration of the rule base via adding, changing or refining the knowledge which might easily lead to incompleteness and contradictions.
- *Anomalies* are considered as symptoms of genuine errors, i.e. they man not necessarily represent problems in themselves.

Much work has been done to establish and classify the nature of errors and anomalies that may be present in rule bases, see e.g. the taxonomy of anomalies from Preece and Shinghal [5]. We briefly review the notions that are commonly used in the literature [6, 7], which range from semantic checks for consistency and completeness to structural

checks for redundancy, relevance and reachability:
1. *Consistency*: No conflicting conclusions can be made from a set of valid input data. The common definition of consistency is that two rules or inferences are inconsistent if they succeed at the same knowledge state, but have conflicting results. Several special cases of inconsistent rules are considered in literature such as:
- *self-contradicting rules* and *self-contradicting rule chains*, e.g. $p \wedge q \rightarrow \neg p$
- *contradicting rules* and *contradicting rule chains*, e.g. $p \wedge q \rightarrow s$ and $p \wedge q \rightarrow \neg s$
Note that the first two cases of self-contradiction are not consistent in a semantic sense and can equally be seen as redundant rules, since they can be never concluded.
2. *Correctness/Soundness*: No invalid conclusions can be inferred from valid input data, i.e. a rule base is correct when it holds for any complete model M, that the inferred output from valid inputs via the rule base are true in M. This is closely related to *soundness* which checks that the intended outputs indeed follows from the valid input. Note, that in case of partial models with only partial information this means that all possible partial models need to be verified instead of only the complete models. However, for monotonic inferences these notions coincide and a rule base which is sound is also consistent.
3. *Completeness:* No valid input information fails to produce the intended output conclusions, i.e. completeness relates to gaps (incomplete knowledge) in the knowledge base. The iterative process of building large rule bases where rules are tested, added, changed and refined obviously can leave gaps such as missing rules in the knowledge base. This usually results in intended derivations which are not possible. Typical sources of incompleteness are missing facts or rules which prevent intended conclusions to be drawn. But there are also other sources. A KB having too many rules and too many input facts negatively influences performance and may lead to incompleteness due to termination problems or memory overflows. Hence, superfluous rules and non-terminating rule chains can be also considered as completeness problems, e.g.:
- *Unused rules and facts*, which are never used in any rule/query derivation (backward reasoning) or which are unreachable or dead-ends (forward reasoning).
- *Redundant rules* such as identical rules or rule chains, e.g. $p \rightarrow q$ and $p \rightarrow q$.
- *Subsumed rules*, a special case of redundant rules, where two rules have the same rule head but one rule contains more prerequisites (conditions) in the body, e.g. $p \wedge q \rightarrow r$ and $p \rightarrow r$.
- *Self-contradicting rules*, such as $p \wedge q \wedge \neg p \rightarrow r$ or simply $p \rightarrow \neg p$, which can never succeed.
- *Loops* in rules of rule chains, e.g. $p \wedge q \rightarrow q$ or tautologies such as $p \rightarrow p$.

3 Homogeneous Integration of Test Cases and Integrity Constraints into Logic Programs

The relevance of V&V of rule bases and LPs has been recognized in the past (see section 2 and 7) and most recently also in the context of policy explanations [8]. The majority of these approaches rely on debugging the derivation trees and giving explanations (e.g. via spy and trace commands) or transforming the program into other more abstract representation structures such as graphs, petri nets or algebraic structures which are then analyzed for inconsistencies. Typically, the definition of an inconsistency, error or anomaly (see section 2) is then given in the language used for analyzing the LP, i.e. the V&V information is not expressed in the same representation language as the rules. This is in strong contrast to the way people would like to engineer, manage and maintain rule-based policies and systems. Different skills for writing the formalized specifications and for analyzing them are needed as well as different systems for reasoning with rules and for V&V. Moreover, the used V&V methodologies (e.g. model checking or graph theory) are typically much more complicated than the rule-based programs. In fact, it turns out that even writing rule-based systems that are useful in practice is already of significant complexity, e.g. due to non-monotonic features or different negations, and that simple methods are needed to safeguard the engineering and maintenance process w.r.t. V&V&I. Therefore, what policy engineers and practitioners would like to have is an "easy-to-use" approach that allows representing rules and tests in the same homogeneous representation language, so that they can be engineered, executed, maintained and interchanged together using the same underlying syntax, semantics and execution/inference environment. In this section we elaborate on this homogeneous integration approach based on the common "denominator": *extended logic programming.*

In the following we use the standard LP notation with an ISO Prolog related scripting syntax called Prova [9] and we assume that the reader is familiar with logic programming techniques [10]. For the semantics of the KB we adapt a rather general definition [11] of LP semantics, because our test-driven approach is intended to be general and applicable to several logic classes / rule languages (e.g. propositional, DataLog, normal, extended) in order to fulfill the different KR needs of particular policy/contract projects (w.r.t expressiveness and computational complexity which are in a trade-off relation to each other). In particular, as we will show in section 5, TCs can be also used to verify the possible unknown semantics of a target inference service in a open environment such as the (Semantic) Web and test the correct execution of an interchanged policy/contract in the target environment.

- A semantics $SEM(P)$ of a LP P is proof-theoretically defined as a set of literals that are derivable from P using a particular derivation mechanisms, such as linear SLD(NF)-resolution variants with negation-as-finite-failure rule or non-linear tabling approaches such as SLG resolution. Model-theoretically, a semantics $SEM(P)$ of a program P is a subset of all models of P: $MOD(P)$. In this paper in most cases a subset of the (3-valued) Herbrand-models of the language of L_P: $SEM(P) \subseteq MOD_{Herb L_P}(P)$. Associated to SEM(P) are two entailment relations:

1. sceptical, where the set of all atoms or default atoms are true in all models of SEM(P)

2. credulous, where the set of all atoms or default atoms are true in at least one model of SEM(P)

- A semantics SEM' extends a semantics SEM denoted by $SEM' \geq SEM$, if for all programs P and all atoms l the following holds: $SEM(P) \models l \Rightarrow SEM'(P) \models l$, i.e. all atoms derivable from SEM with respect to P are also derivable from SEM', but SEM' derives more true or false atoms than SEM. The semantics SEM' is defined for a class of programs that strictly includes the class of programs with the semantics SEM. SEM' coincides with SEM for all programs of the class of programs for which SEM is defined.

In our ContractLog reference implementation we mainly adopt the sceptical viewpoint on extended LPs and apply an extended linear SLDNF variant as procedural semantics which has been extended with explicit negation, goal memoization and loop prevention to overcome typical restrictions of standard SLDNF and compute WFS (see ContractLog inference engine).

The general idea of TCs in SE is to predefine the intended output of a program or method and compare the intended results with the derived results. If both match, the TC is said to capture the intended behavior of the program/method. Although there is no 100% guarantee that the TCs defined for V&V of a program exclude every unintended results of the program, they are an easy way to approximate correctness and other SE-related quality goals (in particular when the TCs and the program are refined in an evolutionary, iterative process with a feedback loop). In logic programming we think of a LP as formalizing our knowledge about the world and how the world behaves. The world is defined by a set of models. The rules in the LP constrain the set of possible models to the set of models which satisfy the rules w.r.t the current KB (actual knowledge state). A query Q to the LP is typically a conjunction of literals (positive or negative atoms) $G_1 \wedge .. \wedge G_n$, where the literals G_i may contain variables. Asking a query Q to the LP then means asking for all possible substitutions θ of the variables in Q such that $Q\theta$ logically follows from the LP P. The substitution set θ is said to be the answer to the query, i.e. it is the output of the program P. Hence, following the idea of TCs, for V&V of a LP P we need to predefine the intended outputs of P as a set of (test) queries to P and compare it with the actual results / answers derived from P by asking these test queries to P. Obviously, the set of possible models of a program might be quite large (even if many constraining rules exist), e.g. because of a large fact base or infinite functions. As a result the set of test queries needed to test the program and V&V of the actual models of P would be in worst case also infinite. However, we claim that most of the time correctness of a set of rules can be assured by testing a much smaller subset of these models. In particular, as we will see in the next section, in order to be an adequate cover for a LP the tests need to be only a least general instantiation (specialization) of the rules' terms (arguments) which fully investigates and tests all rules in P. This supports our second claim, that V&V of LPs with TC can be almost ever done in reasonable time, due to the fact that the typical test query is a ground query (without variables) which has a small search space (as compared to queries with free variables) and only proves existence of at least one model satisfying it. In analogy to TCs in SE we define a TC as $TC := \{A, T\}$ for a LP P to consists of:

1. a set of possibly empty input *assertions* "A" being the set of temporarily asserted test input facts (and additionally meta test rules - see section 5) defined over the alphabet "L". The assertions are used to temporarily setup the test environment. They can be e.g. used to define test facts, result values of (external) functions called by procedural attachments, events and actions for testing reactive rules or additional meta test rules.
2. a set of one ore more *tests* T. Each test T_i, $i > 0$ consists of:
- a *test query* Q with goal literals of the form $q(t_1, ..t_n)$?, where $Q \in rule(P)$ and $rule(P)$ is the set of literals in the head of rules (since only rules need to be tested)
- a *result* R being either a positive "$true$", negative "$false$" or "$unknown$" label.
- an intended *answer set* θ of expected variable bindings for the variables of the test query Q: $\theta := \{X_1, ..X_n\}$ where each X_i is a set of variable bindings $\{X_i/a_1, .., X_i/a_n\}$. For ground test queries $\theta := \emptyset$.

We write a TC T as follows: $T = A \cup \{Q => R : \theta\}$. If a TC has no assertions we simply write $T = \{Q => R : \theta\}$. For instance, a TC $T1 = \{p(X) => true : \{X/a, X/b, X/b\}, q(Y) => false\}$ defines a TC $T1$ with two test queries $p(X)$? and $q(Y)$?. The query $p(X)$? should succeed and return three answers a,b and c for the free variable X. The query $q(Y)$ should fail. In case we are only interested in the existential success of a test query we shorten the notation of a TC to $T = \{Q => R\}$.

To formulate runtime consistency criteria w.r.t. conflicts which might arise due to knowledge updates, e.g. adding rules, we apply ICs:

An IC on a LP is defined as a set of conditions that the constrained KB must satisfy, in order to be considered as a consistent model of the intended (real-world domain-specific) model. Satisfaction *of an IC is the fulfillment to the conditions imposed by the constraint and* violation *of an IC is the fact of not giving strict fulfillment to the conditions imposed by the constraint, i.e. satisfaction resp. violation on a program (LP) P w.r.t the set of* $IC := \{ic_1, ..ic_i\}$ *defined in P is the satisfaction of each* $ic_i \in IC$ *at each KB state* $P' := P \cup M_i \Rightarrow P \cup M_{i+1}$ *with* $M_0 = \emptyset$, *where* M_i *is an arbitrary knowledge update adding,removing or changing rules or facts to the dynamically extended or reduced KB.*

Accordingly, ICs are closely related to our notion of TCs for LPs. In fact, TCs can be seen as more expressive ICs. From a syntactical perspective we distinguish ICs from TCs, since in our (ContractLog) approach we typically represent and manage TCs as stand-alone LP scripts (module files) which are imported to the KB, whereas ICs are defined as LP functions. Both, internal ICs or external TCs can be used to define conditions which denote a logic or application specific conflict. ICs in ContractLog are defined as a n-ary function $integrity(< operator >, < conditions >)$. We distinguish four types of ICs:

- *Not-constraints* which express that none of the stated conclusions should be drawn.
- *Xor-constraints* which express that the stated conclusions should not be drawn at the same time.
- *Or-constraints* which express that at least one of the stated conclusions must be drawn.
- *And-constraints* which express that all of the stated conclusion must draw.

ICs are defined as constraints on the set of possible models and therefore describe the model(s) which should be considered as strictly conflicting. Model theoretically we attribute a 2-valued truth value (true/false) to an IC and use the defined set of constraints (literals) in an IC as a goal on the program P, by meta interpretation (proof-theoretic semantics) of the integrity functions. In short, the truth of an IC in a finite interpretation I is determined by running the goal G_{IC} defined by the IC on the clauses in P or more precisely on the actual knowledge state of P_i in the KB. If the G_{IC} is satisfied, i.e. there exists at least one model for the sentence formed by the G_{IC}: $P_i \models G_{IC}$, the IC is violated and P is proven to be in an inconsistent state w.r.t. IC: IC is violated resp. P_i violates integrity iff for any interpretation I, $I \models P_i \rightarrow I \models G_{IC}$. We define the following interpretation for ICs:

And $and(C_1, .., C_n)$: $P_i \models (notC_1 \vee .. \vee notC_n)$ if exists $i \in 1, .., n, P_i \models not\ C_i$

Not: $not(C_1, .., C_n)$: $P_i \models (C_1 \vee .. \vee C_n)$ if exists $i \in 1, .., n, P_i \models C_i$

Or: $or(C_1, .., C_n)$: $P_i \models (notC_1 \wedge .. \wedge notC_n$ if for all $i \in 1, .., n, P_i \models not\ C_i$

Xor: $xor(C_1, .., C_n)$: $P_i \models (C_j \wedge C_k)$ if exists $j \in 1, .., n, P_i \models C_j$ and exists $k \in 1, .., n, P_i \models C_k$ with $C_j \neq C_k$ and $C_j \in C$, $C_k \in C$

$C := \{C_1, .., C_n\}$ are positive or negative n-ary atoms which might contain variables; *not* is used in the usual sense of default negation, i.e. if a constraint literal can not be proven true, it is assumed to be false. If there exists a model for a IC goal (as defined above), i.e. the "integrity test goal" is satisfied $P_i \models G_{IC}$, the IC is assigned true and hence integrity is violated in the actual knowledge/program state P_i.

4 Declarative Test Coverage Measurement

Test coverage is an essential part of the feedback loop in the test-driven engineering process. The coverage feedback highlights aspects of the formalized policy/contract specification which may not be adequately tested and which require additional testing. This loop will continue until coverage of the intended models of the formalized policy specification meets an adequate approximation level by the TC resp. test suites (TS) which bundle several TCs. Moreover, test coverage measurements helps to avoid atrophy of TSs when the rule-based specifications are evolutionary extended. Measuring coverage helps to keep the tests up to a required level if new rules are added or existing rules are removed/changed. However, conventional testing methods for imperative programming languages rely on the control flow graph as an abstract model of the program or the explicitly defined data flow and use coverage measures such as branch or path coverage. In contrast, the proof-theoretic semantics of LPs is based on resolution with unification and backtracking, where no explicit control flow exists and goals are

used in a refutation attempt to specialize the rules in the declarative LP by unifying them with the rule heads. Accordingly, building upon this central concept of unification *a test covers a logic program P, if the test queries (goals) lead to a least general specialization of each rule in P, such that the full scope of terms (arguments) of each literal in each rule is investigated by the set of test queries.*
Inductively deriving general information from specific knowledge is a task which is approached by inductive logic programming (ILP) techniques which allow computing the least general generalization (lgg), i.e. the most specific clause (e.g. w.r.t. theta subsumption) covering two input clauses. A lgg is the generalization that keeps an generalized term t (or clause) as special as possible so that every other generalization would increase the number of possible instances of t in comparison to the possible instances of the lgg. Efficient algorithms based on syntactical anti-unification with θ-subsumption ordering for the computation of the (relative) lgg(s) exist and several implementations have been proposed in ILP systems such as GOLEM, or FOIL. θ-subsumption introduces a syntactic notion of generality: A rule (clause) r (resp. a term t) θ-subsumes another rule r', if there exists a substitution θ, such that $r \subseteq r'$, i.e. a rule r is *as least as general as* the rule r' ($r \leq r'$), if r θ-subsumes r' resp. *is more general than* r' ($r < r'$) if $r \leq r'$ and $r' \nleq r$. (see e.g. [14]) In order to determine the level of coverage the specializations of the rules in the LP under test are computed via specializing the rules with the test queries by standard unification. Then via generalizing these specializations under θ-subsumption ordering, i.e. computing the lggs of all successful specializations, a reconstruction of the original LP is attempted. The number of successful "recoverings" then give the level of test coverage, i.e. the level determines those statements (rules) in a LP that have been executed/investigated through a test run and those which have not. In particular, if the complete LP can be reconstructed via generalization of the specialization then the test fully covers the LP. Formally we express this as follows:
Let T be a test with a set of test queries $T := \{Q_1?, .., Q_n?\}$ for a program P, then T is a cover for a rule $r_i \in P$, if the $lgg(r_i') \simeq r_i$ under $\theta - subsumption$, where $\simeq$ is an equivalence relation denoting variants of clauses/terms and the r_i' are the specializations of r_i by a query $Q_i \in T$. It is a cover for a program P, if T is a cover for each rule $r_i \in P$. With this definition it can be determined whether a test covers a LP or not. The coverage measure for a LP P is then given by the number of covered rules r_i divided by the number k of all rules in P:

$$cover_P(T) : -\frac{\sum_{i=1}^{k} cover_{r_i}(T)}{k}$$

For instance, consider the following simplified business policy P:

```
discount(Customer, 10%) :- gold(Customer).
gold(Customer) :- spending(Customer, Value) , Value > 3000.
spending('Moor',5000). spending('Do',4000). %facts
```

Let $T = \{discount('Moor', 10\%)? => true, discount('Do', 10\%)? => true$ be a test with two test queries. The set of directly derived specializations by applying this tests on P are:

```
discount('Moor',10%) :- gold('Moor').
discount('Do',10%) :- gold('Do').
```

The computed lggs of this specializations are:

```
discount(Customer,10%) :- gold(Customer).
```

Accordingly, the coverage of P is 50%. We extend T with the additional test goals: $\{gold('Moor')? => true, gold('Do')? => true)?\}$. This leads to two new specializations:

```
gold('Moor') :- spending('Moor',Value) , Value > 3000.
gold('Do') :- spending('Do',Value) , Value > 3000.
```

The additional lggs are then:

```
gold(Customer) :- spending(Customer, Value) , Value > 3000.
```

T now covers P, i.e. coverage = 100%.

The coverage measure determines how much of the information represented by the rules is already investigated by the actual tests. The actual lggs give feedback how to extend the set of test goals in order to increase the coverage level. Moreover, repeatedly measuring the test coverage each time when the rule base becomes updated (e.g. when new rules are added) keeps the test suites (set of TCs) up to acceptable testing standards and one can be confident that there will be only minimal problems during runtime of the LP because the rules do not only pass their tests but they are also well tested. In contrast to other computations of the lggs such as implication (i.e. a stronger ordering relationship), which becomes undecidable if functions are used, θ-subsumption has nice computational properties and it works for simple terms as well as for complex terms with or without negation, e.g. $p() : -q(f(a))$ is a specialization of $p : -q(X)$. Although it must be noted that the resulting clause under generalization with θ-subsumption ordering may turn out to be redundant, i.e. it is possible find an equivalent one which is described more shortly, this redundancy can be reduced and since we are only generalizing the specializations on the top level this reduction is computationally adequate. Thus, θ-subsumption and least general generalization qualify to be the right framework of generality in the application of our test coverage notion.

5 Test-driven V&V of Rule Engines and Rule Interchange

Typical rule-based contracts/policies are managed and maintained in a distributed environment where the rules and data is interchanged over domain boundaries using more or less standardized rule markup interchange formats, e.g. RuleML, SWRL, RBSLA, RIF. The interchanged rules need to be interpreted and executed correctly in the target inference engine which might be provided as an open (Web) service by a third-party provider or a standardization body such as OMG or W3C (see [15]). Obviously, the correct execution of the interchanged LP depends on the semantics of both, the LP and the the inference engine (IE). TCs, which are interchanged together with the LP, can be used to test whether the LP still behaves as intended in the target environment.

To address this issues the IE, the interchanged LP and the provided TCs must reveal their semantics, e.g. by use of explicit meta annotations based on a common vocabulary such as a (Semantic Web) ontology which classifies semantics such as *STABLE* (stable model), *WFS* (well-founded) and relates them to classes of LPs such as *stratified LPs*, *normal LPs*, *extended LPs*. The ontology can then be used to provide additional meta information about the semantics and logic class of the interchanged rules and TCs and find appropriate IEs to correctly and efficiently interpret and execute the LP, e.g. (1) via configuring the target rule engine for a particular semantics in case it supports different ones (see e.g. the configurable ContractLog IE), (2) by executing an applicable variant of several interchanged semantics alternatives of the LP or (3) by automatic transformation approaches which transform the interchange LP into an executable LP. However, we do not believe that each rule engine vendor will annotate its implementation with such meta information, even when there is an official standard

Semantic Web ontology on hand (e.g. released by OMG or W3C). Therefore, means to automatically determine the supported semantics of IEs are needed. As we will show in this section, TCs can be extended to *meta test programs* testing typical properties of well-known semantics and by the combination of succeed and failed meta tests uniquely determine the unknown semantics of the target environment.

A great variety of semantics for LPs and non-monotonic reasoning have been developed in the past decades. For an overview we relate to [11]. In general, there are three ways to determine the semantics (and hence the IE) to be used for execution: (1) by its *complexity and expressiveness class* (which are in a trade-off relation to each other), (2) by its *runtime performance* or (3) by the *semantic properties* it should satisfy. A generally accepted criteria as to why one semantics should be used over another does not exists, but two main competing approaches, namely WFS and STABLE, have been broadly accepted as declarative semantics for normal LPs.

For discussion of the worst case complexity and expressiveness of several classes of LPs we refer to [16]. Based on these complexity results for different semantics and expressive classes of LPs, which might be published in a machine interpretable format (Semantic Web ontology) for automatic decision making, certain semantics might be already excluded to be usable for a particular rule-based policy/contract application. However, asymptotic worst-case results are not always appropriate to quantify performance and scalability of a particular rule execution environment since implementation specifics of an IE such as the use of inefficient recursions or memory-structures might lead to low performance or memory overflows in practice. TCs can be used to measure the runtime performance and scalability for different outcomes of a rule set given a certain test fact base as input. By this certain points of attention, e.g., long computations, loops or deeply nested derivation trees, can be identified and a refactoring of the rule code (e.g. reordering rules, narrowing rules, deleting rules etc.) can be attempted [17]. We call this *dynamic testing* in opposite to *functional testing. Dynamic TCs* with maximum time values (time constraints) are defined as an extension to functional TCs (see section 3): $TC = A \cup \{Q => R : \theta < MS\}$, where MS is a maximum time constraint for the test query Q. If the query was not successful within this time frame the test is said to be failed. For instance, $TC_{dyn} : q(a)? => true < 1000ms$ succeeds iff the test query succeeds and the answer is computed in less than 1000 milliseconds.

To define a meta ontology of semantics and LP classes (represented as an OWL ontology - see [18]) which can be used to meta annotate the interchanged policy LPs, the IEs and the TCs we draw on the general semantics classification theory developed by J. Dix [12, 13]. Typical top-level LP classes are, e.g., definite LPs, stratified LPs, normal LP, extended LPs, disjunctive LPs. Well-known semantics for these classes are e.g., least and supported Herbrand models, 2 and 3-valued COMP, WFS, STABLE, generalized WFS etc. Given the information to which class a particular LP belongs, which is its intended semantics and which is the de facto semantics of the target IE, it is straightforward to decide wether the LP can be executed by the IE or not. In short, a LP can not be executed by an IE, if the IE derives less literals than the intended SEM for which the LP was design for would do, i.e. $SEM'(IE) \geq SEM(P)$ or the semantics implemented by the IE is not adequate for the program, i.e. $SEM'(IE) \neq SEM(P)$. This information can be give by meta annotations, e.g., *class:* defines the class of the LP / IE; *semantics:* defines the semantics of the LP / IE; *syntax:* defines the rule language syntax.

In the context of rule interchange with open, distributed IEs, which might be provided as public services, an important question is, wether the IE correctly implements a semantics. *Meta TCs* can be used for V&V of the interchanged LP in the target environment and therefore establish trust to this service. Moreover, meta TCs checking general properties of semantics can be also used to verify and determine the semantics of the target IE even in case when it is unknown (not given by meta annotations). Kraus et al. [19] and Dix [12, 13] proposed several weak and structural (strong) properties for arbitrary (non-monotonic) semantics, e.g.:

Strong Properties
- *Cumulativity*: If $U \subseteq V \subseteq SEM_P^{scept}(U)$, then $SEM_P^{scept}(U) = SEM_P^{scept}(V)$, where U and V are are sets of atoms and SEM_P^{scept} is an arbitrary sceptical semantics for the program P, i.e. if $a \mid\sim b$ then $a \mid\sim c$ iff $(a \wedge b) \mid\sim c$.
- *Rationality*: If $U \subseteq V, V \cap \{A : SEM_P^{scept}(U) \models \neg A\} = \emptyset$, then $SEM_P^{scept}(U) \subseteq SEM_P^{scept}(V)$.

Weak Properties
- *Elimination of Tautologies*: If a rule $a \leftarrow b \wedge not\ c$ with $a \cap b = \emptyset$ is eliminated from a program P, then the resulting program P' is semantically equivalent: $SEM(P) = SEM(P')$. a,b,c are sets of atoms: $P \mapsto P'$ iff there is a rule $H \leftarrow B \in P$ such that $H \in B$ and $P' = P \setminus \{H \leftarrow b\}$
- *Generalized Principle of Partial Evaluation (GPPE)*: If a rule $a \leftarrow b \wedge not\ c$, where b contains an atom B, is replaced in a program P' by the n rules $a \cup (a^i - B) \leftarrow ((b - B) \cup b^i) \wedge not\ (c \cup c^i)$, where $a^i \leftarrow b^i \wedge not\ c^i (i = 1, ..n)$ are all rules for which $B \in a^i$, then $SEM(P) = SEM(P')$
- *Positive/Negative Reduction*: If a rule $a \leftarrow b \wedge not\ c$ is replaced in a program P' by $a \leftarrow b \wedge not\ (c - C)$ (C is an atom), where C appears in no rule head, or a rule $a \leftarrow b \wedge not\ c$ is deleted from P, if there is a fact a' in P such that $a' \subseteq c$, then $SEM(P) = SEM(P')$:
1. Positive Reduction: $P \mapsto P'$ iff there is a rule $H \leftarrow B \in P$ and a negative literal $not\ B \in B$ such that $B \ni HEAD(P)$ and $P' = (P \setminus \{H \leftarrow B\}) \cup \{H \leftarrow (B \setminus \{notB\})\}$
2. Negative Reduction: $P \mapsto P'$ iff there is a rule $H \leftarrow B \in P$ and a negative literal $not\ B \in B$ such that $B \in FACT(P)$ and $P' = (P \setminus \{H \leftarrow B\})$
- *Elimination of Non-Minimal Rules / Subsumption*: If a rule $a \leftarrow b \wedge not\ c$ is deleted from a program P if there is another rule $a' \leftarrow b' \wedge not\ c'$ such that $a' \subseteq a, b' \subseteq b, c' \subseteq c$, where at least one $\subseteq$ is proper, then $SEM(P) = SEM(P')$: $P \mapsto P'$ iff there are rules $H \leftarrow B$ and $H \leftarrow B' \in P$ such that $B \subset B'$ and $P' = P \setminus \{H \leftarrow B'\}$
- *Consistency*: $SEM(P) = \emptyset$ for all disjunctive LPs
- *Independence*: For every literal L, L is true in every $M \in SEM(P)$ iff L is true in every $M \in SEM(P \cup P')$ provided that the language of P and P' are disjoint and L belongs to the language of P
- *Relevance*: The truth value of a literal L with respect to a semantics $SEM(P)$, only depends on the subprogram formed from the *relevant rules* of P ($relevant(P)$) with respect to L: $SEM(P)(L) = SEM(relevant(P, L))(L)$

The basic idea to apply these properties for the V&V as well as for the *automated determination* of the semantics of arbitrary LP rule inference environments is, to translate known *counter examples* into meta TCs and apply them in the target IE. Such counter examples which show that certain semantics do not satisfy one or more of the general properties, have been discussed in literature. To demonstrate this approach we will now give an examples derived from [12, 13]. For a more detailed discussion of this meta test case approach and more examples see [18]:

```
Example: STABLE is not Cautious
P:  a <- neg b        P': a <- neg b
    b <- neg a            b <- neg a
    c <- neg c            c <- neg c
    c <- a                c <- a
                          c
T:{a?=>true,c?=>true}
```

```
STABLE(P) has {a, neg b, c} as its only stable model and hence it derives ’a’ and ’c’, i.e.
’T’ succeeds. By adding the derived atom ’c’ we get another model for P’ {neg a, b, c}, i.e.
’a’ can no longer derived (i.e. ’T’ now fails) and cautious monotonicity is not satisfied.

Example: STABLE does not satisfy Relevance
P:  a <- neg b           P’: a <- neg b
                             c <- neg c
T:={a?=>true}

The unique stable model of ’P’ is {a}. If the rule ’c <- neg c’ is added, ’a’ is no longer
derivable because no stable model exists. Relevance is violated, because the truth value of
’a’ depends on atoms that are totaly unrelated with ’a’.
```

The first *"positive"* meta TC is used to verify if the (unknown) semantics implemented by the IE will provide the correct answers for this particular meta test program P. The "negative" TC P' is then used to evaluate if the semantics of the IE satisfies the property under tests. Such sets of meta TCs provide us with a tool for determining an "adequate" semantics to be used for a particular rule-based policy/contract application. Moreover, there are strong evidences that by taking all properties together an arbitrary semantics might be uniquely determined by the set of satisfied and unsatisfied properties, i.e. via applying a meta TS consisting of adequate meta TCs with typical counter examples for these properties in a IE, we can uniquely determine the semantics of this IE. Table 1 (derived from [12, 13]) specifies for common semantics the properties that they satisfy.

Table 1. Table (General Properties of Semantics)

Semantics	Class	Cumul.	Rat.	Taut.	GPPE	Red.	Non-Min.	Rel.	Cons.	Indep.
COMP	Normal	-	●	-	●	●	●	-	-	-
COMP$_3$	Normal	●	●	-	●	●	●	-	-	-
WFS	Normal	●	●	●	●	●	●	●	●	●
STABLE	Normal	-	●	●	●	●	●	-	-	-
WGCWA	Pos. Disj.	-	●	-	●	●	-	●	●	●
CGWA	Strat. Disj.	●	-	●	●	●	●	●	●	●
PERFECT	Strat.Disj.	●	-	●	●	●	●	-	●	●

The semantic principles described in this section are also very important in the context of applying *refactorings* to LPs. In general, a refactoring to a rule base should optimize the rule code without changing the semantics of the program. Removing tautologies or non-minimal rules or applying positive/negative reductions are typically applied in rule base refinements using refactorings [17] and the semantics equivalence relation between the original and the refined program defined for this principles is therefore an important prerequisite to safely apply a refactoring of this kind.

6 Integration into Testing Frameworks and RuleML

We have implemented the test drive approach in the ContractLog KR [18]. The ContractLog KR [1] is an expressive and efficient KR framework developed in the RBSLA project [3] and hosted at Sourceforge for the representation of contractual rules, policies and SLAs implementing several logical formalisms such as event logics, defeasible logic, deontic logics, description logic programs in a homogeneous LP framework as meta programs. TCs in the ContractLog KR are homogeneously integrated into LPs and are written in an extended ISO Prolog related scripting syntax called Prova [9]. A TC script consists of (1) a unique ID denoted by the function *testcase(ID)*, (2) optional input assertions such as input facts and test rules which are added temporarily to the KB as partial modules by expressive ID-based update functions, (3) a positive meta test rule defining the test queries and variable bindings *testSuccess(Test Name,Optional Message for Junit)*, (4) a negative test rule *testFailure(Test Name,Message)* and (5) a *runTest* rule.

```
% testcase oid
testcase("./examples/tc1.test").
% assertions via ID-based updates adding one rule and two facts
:-solve(update("tc1.test","a(X):-b(X). b(1). b(2).")).
% positive test with success message for JUnit report
testSuccess("test1","succeeded"):- testcase(./examples/tc1.test),testQuery(a(1)).
% negative test with failure message for Junit report
testFailure("test1","can not derive a"):- not(testSuccess("test1",Message)).
% define the active tests - used by meta program
runTest("./examples/tc1.test"):-testSuccess("test 1",Message).
```

A TC can be temporarily loaded resp. removed to/from the KB for testing purposes, using expressive ID-based update functions for dynamic LPs [18]. The TC meta program implements various functions, e.g., to define positive and negative test queries (*testQuery, testNotQuery, testNegQuery*), expected answer sets (variable bindings: *testResults*) and quantifications on the expected number of result (*testNumberOfResults*). It also implements the functions to compute the clause/term specializations (*specialize*) and generalizations (*generalize*) as well as the test coverage (*cover*). To proof integrity constraints we have implemented another LP meta program in the ContractLog KR with the main test axioms *testIntegrity*() and *testIntegrity*(*Literal*). The first integrity test is useful to verify (test logical integrity) and validate (test application/domain integrity) the integrity of the actual KB against all ICs in the KB. The second integrity test is useful to hypothetically test an intended knowledge update, e.g. test wether a conclusion from a rule (the literal denotes the rule head) will lead to violations of the ICs in the KB. Similar sets of test axioms are provided for temporarily loading, executing and unloading TCs from external scripts at runtime.

To become widely accepted and useable to a broad community of policy engineers and practitioners existing expertise and tools in traditional SE and flexible information system (IS) development should be adapted to the declarative test-driven programming approach. Well-known test frameworks like JUnit facilitate a tight integration of tests into code and allow for automated testing and reporting in existing IDEs such as eclipse via automated Ant tasks. The RBSLA/ ContractLog KR implements support for JUnit based testing and test coverage reporting where TCs can be managed in test suites (represented as LP scripts) and automatically run by a JUnit Ant task. The ContractLog distribution comes with a set of functional-, regression-, performance- and meta-TCs for the V&V of the inference implementations, semantics and meta programs of the ContractLog KR.

To support distributed management and rule interchange we have integrated TCs into RuleML (RuleML 0.9). The Rule Markup Language (RuleML) is a standardization initiative with the goal of creating an open, producer-independent XML/RDF based web language for rules. The Rule Based Service Level Agreement markup language (RBSLA) [2] which has been developed for serialization of rule based contracts, policies and SLAs comprises the TC layer together with several other layers extending RuleML with expressive serialization constructs, e.g. defeasible rules, deontic norms, temporal event logics, reactive ECA rules. The markup serialization syntax for TSs / TCs includes the following constructs given in EBNF notation, i.e. alternatives are separated by vertical bars (|); zero to one occurrences are written in square brackets ([]) and zero to many occurrences in braces ({}).:

```
assertions ::= And
test ::= Test | Query
message ::= Ind | Var
TestSuite ::= [oid,] content | And
TestCase ::= [oid,] {test | Test,}, [assertions | And]
Test ::= [oid,] [message | Ind | Var,] test | Query, [answer | Substitutions]
Substitutions ::= {Var, Ind | Cterm}

Example:

<TestCase @semantics="semantics:STABLE" class="class:Propositional">
    <Test @semantics="semantics:WFS" @label="true">
        <Ind>Test 1</Ind><Ind>Test 1 failed</Ind>
        <Query>
            <And>
                <Atom><Rel>p</Rel></Atom>
                <Naf><Atom><Rel>q</Rel></Atom></Naf>
            ...
</TestCase>
```

The example shows a test case with the test: $test1 : \{p => true, not\ q => true\}$.

7 Related Work and Conclusion

V&V of KB systems and in particular rule based systems such as LPs with Prolog interpreters have received much attention from the mid '80s to the early '90s, see e.g. [6]. Several V&V methods have been proposed, such as methods based on *operational debugging* via instrumenting the rule base and exploring the execution trace, *tabular methods*, which pairwise compare the rules of the rule base to detect relationships among premises and conclusions, methods based on *formal graph theory* or *Petri Nets* which translate the rules into graphs or Petri nets, methods based on *declarative debugging* which build an abstract model of the LP and navigate through it or methods based on *algebraic interpretation* which transform a KB into an algebraic structure, e.g. a boolean algebra which is then used to verify the KB. As discussed in section 1 most of this approaches are inherently complex and are not suited for the policy resp. contract domain. Much research has also been directed at the automated refinement of rule bases [17], and on the automatic generation of test cases. There are only a few attempts addressing test coverage measurement for test cases of backward-reasoning rule based programs [22, 23].

Test cases for rule based policies are particular well-suited when policies/contracts grow larger and more complex and are maintained, possibly distributed and interchanged, by different people. In this paper we have attempted to bridge the gap between the test-driven techniques developed in the Software Engineering community, on one hand, and the declarative rule based programming approach for engineering high level policies such as SLAs, on the other hand. We have elaborated on an approach using logic programming as a common basis and have extended this test-driven approach with the notion of test coverage, integrity tests, functional, dynamic and meta tests for the V&V&I of inference environments in a open distributed environment such as the (Semantic) Web. In addition to the homogeneous integration of test cases into LP languages we have introduce a markup serialization as an extension to RuleML which, e.g. facilitates rule interchange. We have implemented our approach in the ContractLog KR [1] which is based on the Prova open-source rule environment [9] and applied the agile test-driven values and practices successfully in the rule based SLA (RBSLA) project for the development of complex, distributed SLAs [3]. Clearly, test cases and test-driven development is not a replacement for good programming practices and rule code review. However, the presence of test cases helps to safeguard the life cycle of policy/contract rules, e.g. enabling V&V at design/development time but also dynamic testing at runtime. In general, the test-driven approach follows the well-known 80-20 rule, i.e. increasing the approximation level of the intended semantics of a rule set (a.k.a. test coverage) by finding new adequate test cases becomes more and more difficult with new tests delivering less and less incrementally. Hence, under a cost-benefit perspective one has to make a break-even point and apply a not too defensive development strategy to reach practical levels of rule engineering and testing in larger rule based policy or contract projects.

References

1. A. Paschke, M. Bichler. Knowledge Representation Concepts for Automated SLA Management, Int. Journal of Decision Support Systems, to appear 2007.
2. A. Paschke. RBSLA - A declarative Rule-based Service Level Agreement Language based on RuleML, Int. Conf. on Intelligent Agents, Web Technology and Internet Commerce, Vienna, Austria, 2005.
3. A. Paschke. RBSLA: Rule-based Service Level Agreements. http://ibis.in.tum.de/staff/paschke/rbsla/index.htm or https://sourceforge.net/projects/rbsla.
4. A.J. Gonzales, V. Barr. Validation and verification of intelligent systems. *Journal of Experimental and Theoretical AI.* 2000.
5. A.D. Preece and Shinghal R. Foundations and applications of Knowledge Base Verification. *Int. J. of Intelligent Systems.* Vol. 9, pp. 683-701, 1994.
6. G. Antoniou, F. v. Harmelen, R Plant, and J Vanthienen. Verification and validation of knowledge-based systems - report on two 1997 events. AI Magazine, 19(3):123126, Fall 1998.
7. A. Preece. Evaluating verification and validation methods in knowledge engineering. University of Aberdeen, 2001.
8. P. Bonatti, D. Olmedilla, and J Peer. Advanced policy explanations. In 17th European Conference on Artificial Intelligence (ECAI 2006), Riva del Garda, Italy, Aug-Sep 2006. IOS Press.
9. A. Kozlenkov, A. Paschke, M. Schroeder, Prova - A Language for Rule Based Java Scripting, Information Integration, and Agent Programming. http://prova.ws., 2006.
10. J.W. Lloyd. Foundations of Logic Programming. 1987, Berlin: Springer.
11. J. Dix. Semantics of Logic Programs: Their Intuitions and Formal Properties. An Overview. In Andre Fuhrmann and Hans Rott, editors, Logic, Action and Information – Essays on Logic in Philosophy and Artificial Intelligence, pages 241–327. DeGruyter, 1995.
12. J. Dix. A Classification-Theory of Semantics of Normal Logic Programs: I. Strong Properties," Fundamenta Informaticae XXII(3) pp. 227-255, 1995.
13. J. Dix. A Classification-Theory of Semantics of Normal Logic Programs: II. Weak Properties. Fundamenta Informaticae, XXII(3):257-288, 1995.
14. G.D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5, 1970.
15. A. Paschke, J. Dietrich and H. Boley. W3C RIF Use Case: Rule Interchange Through Test-Driven Verification and Validation. http://www.w3.org/2005/rules/wg/wiki/Rule_Interchange_Through_Test-Driven_Verification_and_Validation, 2005.
16. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. IEEE Conference on Computational Complexity, pages 82–101, Ulm, Germany, 1997.
17. J. Dietrich and A. Paschke. On the test-driven development and validation of business rules. Int. Conf. ISTA'2005, 23-25 May, 2005, Palmerston North, New Zealand, 2005.
18. A. Paschke. The ContractLog Approach Towards Test-driven Verification and Validation of Rule Bases - A Homogeneous Integration of Test Cases and Integrity Constraints into Dynamic Logic Programs and Rule Markup Languages (RuleML), IBIS, TUM, Technical Report 10/05.
19. S. Kraus, D. Lehmann, M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2):167–207, 1990.
20. P. Meseguer. Expert System Validation Through Knowledge Base Refinement. *IJCAI'93*, 1993.
21. C.L. Chang, J.B. Combs, R.A. Stachowitz. A Report on the Expert Systems Validation Associate (EVA). *Expert Systems with Applications*, Vol.1,No.3,pp. 219-230.
22. R. Denney. Test-Case Generation from Prolog-Based Specifications, IEEE Software, vol. 8, no. 2, pp. 49-57, Mar/Apr, 1991.
23. G. Luo, G. Bochmann, B. Sarikaya, and M. Boyer. Control-flow based testing of prolog programs. In Int. Symp. on Software Reliability Enginnering, pp. 104-113, 1992.

Trust Policies for Semantic Web Repositories

Vinicius da S. Almendra[1], Daniel Schwabe[1]

(1) Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio
Rio de Janeiro - Brazil
{almendra,dschwabe}@inf.puc-rio.br

Abstract. The increasing reliance on information gathered from the Web and other Internet technologies (P2P networks, e-mails, blogs, wikis, etc.) raises the issue of trust. Trust policies are needed to filter out untrustworthy information. This filtering task can be leveraged by the increasing availability of Semantic Web metadata that describes the information retrieved. It is necessary, however, to adequately model the concept of trustworthiness; otherwise one may end up with operational trust measures that lack a clear meaning. It is also important to have a path from one's trust requirements to concrete trust policies through Semantic Web technologies. This paper proposes a horn logic model for trust policies, grounded on a real-world model of trust that offers justification for trust decisions and controlled trust measurement. We also propose the use of this model to enhance existing Semantic Web repositories with a trust layer.

1 Introduction

One of the great challenges of the Web is the problem of trust. Operational measures of trustworthiness are needed to separate relevant and truthful data from those that are not. However, to be correctly interpreted, these measures must be linked with real-world concepts of trust. They must also meet the trust requirements of their users. Building on the trust concepts found in the work of Gerck [10] and Castelfranchi et al. [7], our work aims to pave the path leading from a user's trust requirements to operational trust policies that can be applied to Semantic Web data, while preserving the correspondence between these policies and the trust requirements we started with. This correspondence is important because it enables the user to find out why a piece of data was found trustful.

We focus on the Semantic Web scenario, where an agent aggregates metadata from various sources and the agent must decide which metadata can be trusted and to what extent. Our contributions include a concept of real-world trust, a model to represent this trust concept in the scenario of interest, a simple model to express trust policies using horn logic, including justification for trust decisions, and an implementation, written in Java and Prolog, for the proposed model. It represents an improvement over our previous works [1, 2], where trust degrees and justification were not present.

In section 2 we describe the scenario in which this work fits in. In section 2 we describe the concept of real-world trust on which our work is based, and a model for trust on Semantic Web, based on this trust concept; we also describe a motivating example. In section 4 we describe a model to express trust policies using horn logic. In section 4.3 we describe the implementation of the trust model using Java and Prolog. In section 6 we present works more closely related to our, discussing some differences. In section 7 we conclude our work and point to future directions.

2 A Motivating Scenario

The scenario we focus on is based on the Semantic Web Publishing scenario [6], the DBin project [17] and the Piggy Bank plugin [14]. They all work with the idea of a local repository of information that aggregates new statements from various sources. The first two also include the idea of trustfulness.

The Semantic Web Publishing scenario has information providers and information consumers. An *information provider* publishes RDF graphs, which contain information and its metadata, such as provenance, publishing date, etc. An *information consumer* gathers these graphs and decides what to do with them, treating these graphs as claims by the information provider, rather than definitive facts. The formal meaning of these claims, that is, what statements about the world are being made, is given by a set of accepted graphs, which is a subset of the graphs the information consumer receives.

The Semantic Web Publishing proposal also enables the user to specify a trust policy, that is, a set of conditions that the received information should meet to be accepted. An example of a policy would be “trust all information about computers that comes from direct friends”.

This scenario can be integrated with the one of DBin’s project, which is a P2P network where people exchange RDF graphs of interest and store all the received graphs in a local database. Filtering can be applied to hide triples that do not match the user’s criteria. The set of visible triples, which we call *accepted triples*, is analogous to the set of accepted graphs described above.

These tools can be integrated, giving rise to the scenario we are working with: an agent that continuously aggregates Semantic Web descriptions from various sources and uses these descriptions, together with trust policies, to decide which other descriptions are to be trusted. Recommender systems, reputation management systems, autonomous agents and Social Semantic Desktops can all be seen as particular instances of this scenario, when they are built on top of Semantic Web technologies.

The integration of Piggy Bank and the Semantic Web Publishing scenario, discussed in [4], exemplifies how trust filtering can be integrated to web browsing.

3 A Model for Trust

3.1 A Concept of Trust

To build a suitable trust model, we start by eliciting attributes of real-world trust, trying to capture its essence. Castelfranchi et al. [7] define trust in the context of multi-agent systems, where agents are endowed with goals. In this context, he asserts that trust is "a mental state, a complex attitude of an agent x towards another agent y about the behavior/action α relevant for the result (goal) *g*. This attitude leads the agent *x* to the decision of *relying on y having the behavior/action* α, in order to achieve the goal *g*.

Gerck [10] presents a definition of trust as "what an observer knows about an entity and can rely upon to a qualified extent". This definition has a close parallel with Castelfranchi's: the observer is the agent who trusts; the entity is the trusted agent; the qualified extent is the behavior/action. Both associate trust with reliance. However, the former definition mentions explicitly the goal-oriented nature of trust, which is an important aspect, as agents lacking goals do not really need trust [7].

From both definitions, we observe that trust implies *reliance*: when an agent trusts something, s/he relies on its truth to achieve some goal without further analysis – even if s/he is running the risk of taking an inappropriate or even damaging action if the object of trust is false.

Trust implies reliance, but not necessarily action. For example, John may trust Mary's bookstore without buying anything there. Nevertheless, if John needs a book and Mary offers it under good sale conditions – price, placement, payment etc. –, John *will* buy the book *without further questions*. At the same time, he may refuse to buy the same book under better sale conditions at a bookstore that he does not trust[1]. So, the trust attitude entails a "potential" reliance on the *object* of trust.

We may also ask what the object of trust is. In this case, it could be described as the statement "Mary's bookstore is good". John *believes* this and will act upon it when needed. Using the definition of Gerck, John *knows* that "Mary's bookstore is good" is true and relies on this.

There is another question to be considered: how did John *decide* to trust Mary's bookstore? This is the problem of *justification* [10]. Castelfranchi et al. [7] ground the trust decision on the beliefs of the trusting agent. In our example, one reason John may have decided to trust Mary's bookstore is because he believes Mary is an honest and competent person, and that the business runs under her strict control. If one of these beliefs were absent, then John might not trust Mary's bookstore, according to these premises. Notice that this does not preclude John from trusting Mary's bookstore for other reasons besides this one.

The problem is not solved yet, as we may ask where these beliefs come from. John relies on those beliefs to take a decision (in this case, the decision to trust Mary's bookstore), which characterizes *John's trust on those beliefs*. So, trust decisions may be *chained*: to trust Mary's bookstore, John also has to trust that she is competent and honest. Nevertheless, these trust decisions do not need to be simultaneous: John may have decided to trust Mary's competence many years before she had a bookstore.

There are certain kinds of beliefs widely used to justify trust. One of these is the *self-trust belief*: a person normally trusts facts that are evident to (directly observed by) him. *Provenance belief* is also an important one: when deciding the truthfulness of a statement, one of the first questions is *who* stated it. In fact, the word *statement* implies a provenance: a statement has been *stated* by *someone*.

1 Absence of trust is different from *distrust*, which is a *positive evaluation* of negative qualities: one may not trust a stranger, but will almost certainly distrust a liar.

The justification of trust based on beliefs links trust with *belief revision*: if some of the beliefs that justified a trust decision are discredited, this trust may eventually be lost. If John discovers that several friends bought defective books from Mary, the belief about competence could be revised. Then, trust on Mary's bookstore could become unjustified and might be lost. This is a situation where new evidence hampers previously acquired trust, as this trust was based on the assumption that Mary was competent. It might have been a good assumption, but was shown to be false due to *contradictory evidence*. Here we use the underlying assumption that sometimes the absence of a belief (in this case, the belief that some people bought defective books from Mary) is treated as a positive belief (in this case, the belief that no one has ever bought defective books from Mary or, if someone did, it is not relevant to my decision). This is grounded on the assumption that, if something "wrong" happens (that is, something that may impact an agent's decisions), the agent will eventually be informed about it before he can make damaging decisions.

Another characteristic is that trust is *subjective*: different agents may have different beliefs, different goals and require different degrees of justification to trust something. Continuing with the example, Mike might not trust Mary's bookstore, as he believes she is not competent, she does not know Japanese literature well and she does not worry about the tidiness of the bookstore. Here, we face contradictory beliefs and also different demands to consider a bookstore to be trustful. The difference between beliefs held by John and Mike may be due to the goals: John might be an occasional reader, whereas Mike might be an artist interested in Japanese culture. Note that this goes beyond being a matter of opinion: both make decisions and act based on these beliefs.

Trust also changes over time. John may lose his trust on Mary's bookstore even without any change in his beliefs about her. What changes in such cases is the justification required for trustfulness.

It is of common sense that trust is scalable [7], but this apparently conflicts with the concept of trust as a binary decision (to rely or not). Following Gerck's reasoning [10], the scalability of trust lies is the *degree of justification* required to trust. Stronger trust means stronger evidences. This implies an ordering on the possible justifications for trusting a fact: a better justification assigns a greater trust level to a trusted fact.

3.2 Trust and Semantic Web

At the core of Semantic Web technologies lays RDF (Resource Description Framework) and languages and formalisms based on it, most notably OWL (Web Ontology Language). RDF allows one to describe things using a controlled vocabulary, based on URIs (Uniform Resource Identifiers), through *statements* which are triples in the form (subject, property, object), meaning that the resource identified by *subject* has *property* with value *object*. An RDF document is a set of statements about some reality.

The fact that an agent (human or not) states something does not mean that it is true: one might state that Brazil's capital is Buenos Aires, which is not true (it is Brasília). When an agent uses (that is, relies on) an RDF document, it is implicitly trusting the source of the document on the statements contained in it, which means that s/he trusts every RDF triple in it. Trusting an RDF triple (*S*, *P*, *O*) simply means that s/he believes *S* has the property *P* with value *O*. This trust decision is based on the information contained in the document and on other information available to the agent. From the work of [6], this trust decision may be called *accepting* a triple. After this decision, the triple becomes a belief of the trusting agent.

The decision of whether or not to accept a triple can be modeled as a trust policy which specifies which triples are to be trusted, depending on its components and on other triples the agent has already trusted.

Another important information when dealing with trust is provenance. There are some proposals [6, 9] to add provenance to RDF documents using a fourth element in RDF statements: the context. Although this is not an essential element for trust (one might gather provenance information from other sources), it enhances the expressivenesses of trust policies.

RDF triples may be stored anywhere (in a web page, in an email, in a document in the local file system etc.). In our work we focus on the idea of a repository which holds all triples that are going to be subject to trust evaluation. This solution avoids problems of distributed systems (e.g. lack of network connectivity) and circumscribes the universe of facts used in trust evaluation.

3.3 Outline of a Trust Model

Based on the trust concept formulated in section 3.1 and on Semantic Web concepts discussed in section 3.2, we build an informal model of trust, which comprises the following elements: facts, contexts, knowledge bases, trust policies, trust decisions, justification and trust layers.

A *fact* is a statement about reality, following the semantic of RDF triples. Some authors recognize the need for a fourth element: the *context* [3, 9, 13]. Contexts help define the provenance of the facts (who stated it), the circumstances (date, time, reason, etc.), and, more generally, help situating a fact in order to allow its correct understanding. As these elements are relevant to trust, we will assume the presence of this fourth element. This raises the need for an extension of RDF, such as the named graphs formalism [6]. We will not propose a new extension, but simply assume the availability of contexts and the possibility to obtain provenance and circumstantial information through it.

The set of facts that an agent knows constitutes its *knowledge base*. An *asserted fact* is a known fact and a *trusted fact* is an asserted fact that can be trusted. For example, when someone reads a newspaper, he may augment his knowledge base with several asserted facts, but he may only trust some of them (or none!).

A *trust policy* is a set of rules that the trusting agent uses to test the trustfulness of a fact. Different trusting agents may use different trust policies and, hence, they can make different trust decisions, even when based on the same facts, characterizing the subjective nature of trust. The same trusting agent may change his trust policy in order to match his current goals, which characterizes trust dynamism.

A *trust decision* is the act of testing if an asserted (or inferred according to the domain theory) fact meets a trust policy, that is, a decision to rely on that fact's truthfulness. A trust decision is in fact the process of finding a deduction, which we call a *justification*, that the asserted fact can indeed be trusted. Only trusted facts can be used in a justification. The trust policy specifies trusting agent decides what justifications are acceptable to trust a certain fact. From now on we will call justification any necessary and sufficient set of trusted facts needed to accept an asserted fact as trustworthy. The deduction is done by the trust policy, so a justification is always relative to a specific trust policy and to a specific asserted fact.

Not all justifications grant the same degree of confidence to the trusting agent. Two or three mentions in different web sites might be enough to justify buying a CD from an unknown internet dealer, but would not give enough confidence to buy a car. In both cases, what is in stake is the quality of service offered by the vendor and his/her honesty, but the degree of reliance exhibited in each decision is clearly different. Greater reliance levels demand better justifications.

Given a trust policy and a fact, a *justification level* is an equivalence class of all justifications that are equally good to the trusting agent for that fact with respect to that policy. The set of all justification levels is a partial order: in some situations we say that some justification is better than other, in others situations this statement does not make sense, as one would be comparing apples with oranges. An example of the former is a reputation management system: a person is more reliable than other if its reputation score is greater. An example of the latter is to compare his/her reputation score with the number of citations of papers authored by the other: although both share the idea of refereeing, a direct comparison of these scores does not make sense. A *justification class* is a set of justification levels that form a total order, that is, they are all comparable. Each fact that can have a varying degree of justification may yield a justification class.

A trust policy must assign some justification level to trusted facts. This justification level should reflect some property displayed by the justification that has varying degrees, e.g. the number of positive reviews. When this is not the case, the trust policy may assign the same justification level regardless of the justification facts.

The use of justification levels allows one to rank competing trusted facts in order to choose the one with more evidence. This feature allows the implementation of reputation systems based on trust policies, where the entities of interest are relevant facts from the trusting agent point of view.

The *trust layer* stands between a repository of RDF statements and the application and yields a list of trusted facts together with their justification levels. It does not alter the information in the repository: it just gives trust information under request.

Figure 1 shows the relationship among these concepts. The trust layer augments the knowledge base with justification information, represented by the circles (sets of facts), arcs (justification relationship) and labels on arcs (point out the policy used and the justification level achieved). Colored circles denote trusted facts. Notice that fact 9 was not trusted; policy 1 asserts the trustfulness of fact 4 with justification level A1 due to the presence of the trusted facts 1, 2 and 3. Fact 8 has two different justification levels.

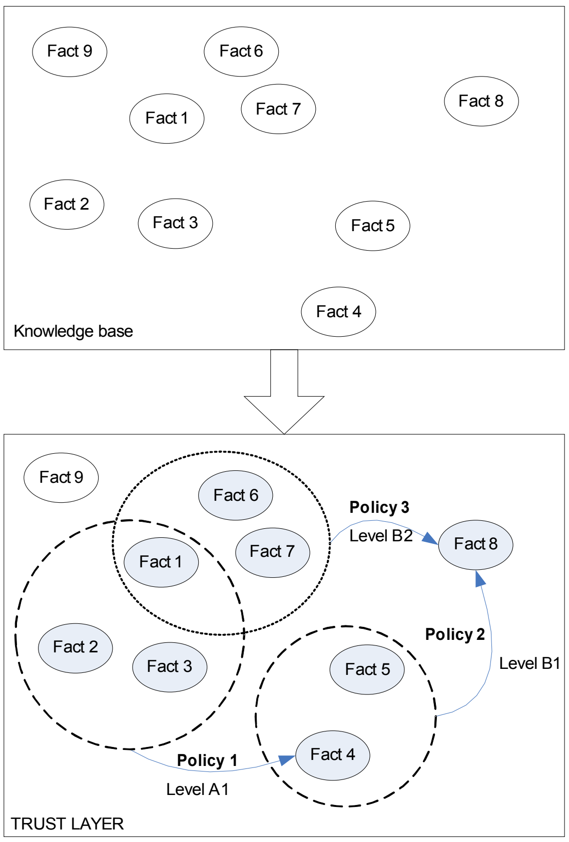


Fig. 1: A Knowledge base and its Trust Layer

3.4 A Motivating Example

Large companies normally have a purchasing department that takes care of buying supplies and purchasing services for the organization. This department deals with a wide variety of products and with many internal clients, that is, people that actually demand the purchasing of new goods or services. It also has a set of suppliers that may change over time, although at a slow rate. As suppliers' performance can vary widely in terms of quality of service, an important function of the purchasing department is to select the right suppliers, given budget restrictions and minimum quality of service requirements.

This task involves dealing of lots of information coming from suppliers, from previous purchases, coming from internal feedback about purchased goods or services, marketing information coming from specialized media etc. Feedback information can be regarded as kind of recommendation. The use of Semantic Web technologies together with trust policies fits well in this scenario, as it uses lots of structured information from various sources and needs to repeatedly assess the trustfulness of statements related to the quality of service and recommendations. The proposed model does not improve over other possible solutions for this problem, but it enables a simpler and cheaper implementation, based on the reuse of existing standards, tools and ontologies like ROR (Resource of a Resource)[2] and eClassOWL[3].

This scenario naturally leads to Semantic Web Services solutions to e-business. Nevertheless, we are assuming a situation were the process is not fully automated: the system makes recommendations based on trust levels, among other factors.

In this scenario, we can devise some trust policies:

- One important factor when buying supplies is the delivery time, so the buyer needs to trust this information. This depends on the past performance of the vendor: if great delays already happened, the information supplied by the vendor is untrustworthy. Here we link data generated when supplies are received to advertised delivery time. The buyer can also choose among vendors with similar delivery times based on the justification level: the chosen vendor is the one whose delivery time has the greatest level of justification.
- Another important factor is the internal feedback from the actual users of supplies. This could be done using a simple ontology, linking a review to a specific purchase, which, in turn, is linked to a vendor. This review should point out some aspect advertised by the vendor that was contradicted (or corroborated) by reality. The system would then assign different justification levels depending on the reviews available.
- The reviews could also be subject of trust evaluation, as sometimes the internal clients may have biases toward some vendors. The reviews of an internal client that are always contradictory to other's reviews might lose their justification level.

The justification for trust is also useful for the buyer. Sometimes s/he might get puzzled with the results of trust evaluation, e.g. when the system assigns a high justification level to a vendor s/he dislikes or when a vendor known to be very good gets a low justification level. The buyer may ask for the facts used to derive trustfulness and may be convinced that the levels assigned were right or may change the trust policies in order to take into account factors that were not contemplated in the trust decision.

4 Building Trust Policies

4.1 A Horn Logic Approach

We will express the model outlined in the previous section, using definite horn clauses to build trust policies. We will later discuss the introduction of "negation as failure" and its consequences.

We assume that the knowledge base of the trusting agent has a *domain theory* that defines all predicates unrelated to the trust model.

Facts will be modeled as RDF triples (*subject, predicate, object)* plus a *context*, similarly to other approaches [4, 7], yielding a quadruple, as discussed above. A context may also be the subject of a fact. We assume that there exists a repository from

2 http://www.rorweb.com/

3 http://www.heppnetz.de/eclassowl/

which all RDF triples come from. There may be some "context" layer that adds contextual information to triples, in the case the repository used does not support this.

A justification class *JC* is represented by a tuple *(J, R)*, where *J* is a nonempty set and *R* is a total order on *J*. The elements of *J* are justification levels. There must be at least one justification class with at least one justification level. The union of all justification levels of all justification classes is called *justification space* (*JS*). The *support set* of a fact is the set of facts that is used to justify trust on it. A fact may have many support sets. A *trust policy* is a predicate with the following signature:

$$p\ (fact, j, supportSet)$$

This predicate means that *fact* can be trusted with justification level *j* given the support set *supportSet*, which is a list of facts.

Trust policies have the following characteristics:

- A fact can be trusted with more than one pair of justification level-support set, as the justification level depends on the support set.
- The policy will find *fact* trustful only if it can deduce the truthfulness of every fact contained in the support set.

The process of finding the trustfulness of an arbitrary fact is recursive and raises the question: which trust policy should be used?

A first solution would be to assume that a single trust policy should decide trustfulness of all facts. So it would use itself recursively to decide trustfulness of the facts contained in the support set. However, as the policy is not tied to specific facts, it does not show clearly the link between the support set and the fact being analyzed. Another solution would be to have specialized policies, each one for a small set of facts. This raises another problem: each policy should be aware of all other policies in order to choose the right one.

We adopt an intermediate solution, assuming the existence of a "general" trust policy that decides trustfulness of all facts, but building this policy through aggregation of smaller (i.e., more restricted) ones. Each component policy must be aware only of the general policy. This general policy identifies itself with the trusting agent's adopted trust policy, as it decides the trustfulness of all facts. From now on we will call it the *root trust policy*. The aggregation can be done using multiple clauses, each one aggregating a new policy, so the root trust policy becomes a disjunction of other policies. For example:

$$rootTrustPolicy(fact, j_1, support1) \leftarrow foafPolicy(fact, j_1, support1)$$
$$rootTrustPolicy(fact, j_2, support2) \leftarrow dcPolicy(fact, j_2, support2)$$
$$rootTrustPolicy(fact, j, support) \leftarrow systemPolicy(fact, j, support)$$

Policies other than the root trust policy will be called elementary trust policies or simply trust policies. In this work we will not go further in the issue of policy composition, as we are investigating the possibility of adopting Bonatti's [5] algebra of policies.

Now we must describe elementary trust policies. An elementary trust policy has the following components: scope definition, a justification, a trustfulness check and a justification level assignment.

The *scope definition* is a set of logical conditions that circumscribes the set of facts that are examined by this policy; facts outside this set are ignored. The simplest restriction is to specify the value of some of the components of the fact, e.g. the predicate. More sophisticated scopes could be ones like “all facts belonging to the FOAF ontology”, “all facts about books written by J. R. R. Tolkien”. The *justification* is composed by a set of conditions that link the fact being examined to other asserted facts (and their components) and put constraints on their values, just like an ordinary database query. The *trustfulness check* consists of checking all facts used in justification for trustfulness using the root trust policy. The *justification level assignment* is the determination of the justification level of the fact being examined. It can be calculated using facts gathered in the justification step. For example, the justification degree of a fact related to a person might depend to the number of people that stated friendship with this person.

In this model, the trusting agent has no “privilege”: it is a source of information like any other. The default policy for self-asserted information would be of accepting it with a high justification level, but nothing prevents the use of policies that give more trust to some other person’s statements than to self-asserted ones.

Trust policies are built around sets of facts. For each relevant (from the trusting agent point of view) set of facts, one or more trust policies can be built. As policies are specified on a per-fact basis, the relationship among them is always done through the root trust policy. This makes the trust policy set easily maintainable, as there are no direct references from one policy to another.

Trust policies may use whatever data is available in the repository. Some policies might demand strong evidences, like PGP signatures or similar mechanisms; other may accept lightweight evidence, like FOAF statements gathered from web pages without further warrants of validity.

Next we show an example of the trust policy “trust the e-mail of a person if s/he is known by a friend of mine”, along with some explanation. Variables are in italics.

foafMboxPolicy(*testFact, justificationLevel, supportSet*) ←	
testFact = (*person1*, foaf:mbox, *email, context1*) AND	Fact to be tested
fact1 = (*person2*, foaf:knows, *person3*, *context2*) AND *fact2* = (*person3*, foaf:mbox_sha1sum, *sha1*, *context2*) AND hasSHA1(*email*, *sha1*) AND *fact3* = (myself, foaf:knows, *person2*, *myContext*) AND *fact4* = (*context2*, dc:creator, *person2*, *context2*) AND *fact5* = (*myContext*, dc:creator, myself, *myContext*) AND	Finds a person that has the same email and is known by a friend of mine
supportSet = {fact1, *fact2*, *fact3*, *fact5*, *fact5}* AND isInRepository(*supportSet*) AND	Tests if the support set is in the knowledge base
checkTrustPolicy(*supportSet*) AND	Checks support set trustfulness
justificationLevel = foaf_mbox	Assigns justification level

This example policy supposes a network of friends that exchange lists of hashed emails, in order to avoid spam without compromising other people’s privacy. Notice the use the predicate hasSHA1, which must be defined elsewhere.

To illustrate the recursive behavior, we can show how a policy that accepts a person as the author (i.e. provenance) of a context would look like:

```
dcCreatorPolicy(testFact, justificationLevel, supportSet) ←
    testFact = (context, dc:creator, person, context) AND
    fact1 = (context, wot:assurance, signature, context) AND
    fact2 = (person, foaf:mbox, email, context) AND
    getPublicKey(email, pubkey, public_key_server) AND
    checkSignature(context, signature, pubkey) AND
    supportSet = {fact1, fact2} AND
    isInRepository(supportSet) AND
    checkTrustPolicy(supportSet) AND
    justificationLevel = default
```

This policy uses the Web of Trust vocabulary and custom predicates in order to check provenance using public key infrastructure.

However, there is a circular dependence: the first policy checks trustfulness of foaf:mbox facts but needs the second one to check dc:creator statements; the second one needs to check foaf:mbox statements, which is done by the first.

The solution lies in the use of justification levels: the second policy does not need a strong verification, as this will be provided by the signature check. So, it can rely on another policy that assigns a default (and lower) trust level to foaf:mbox statements. Notice the use of negation as failure:

```
foafMboxPolicySimple(testFact, justificationLevel, supportSet) ←
    testFact = (person1, foaf:mbox, email, context1) AND
    supportSet = {} AND
    justificationLevel = default
```

This sample presumes that foaf_mbox > default

4.2 Distrust and the Use of Negation

The use of definite (Horn) clauses to express trust policies restricts the use of negation. Intuitively, only positive facts can be tested. However, there may be situations where one understands that the absence of some facts conveys some meaning, as noted in Section 3.1. This can be represented in trust policies by the use of *negation as failure*: the negation of a formula is true iff one cannot prove that formula's truth.

In our model, distrust can be modeled as a trust policy that puts a fact into a unique justification class, different from all others. Then we can use negation as failure to says that a fact can only be trusted if it is *not* trusted within that justification class, that is, distrust always prevents trust, no matter the justification level achieved.

However, the use of negation raises some issues. Semantic Web technologies are based on the Open World Assumption, where the absence of a fact does not imply its falseness. So, when using negation as failure, one assumes a completeness of knowledge that is not warranted by the underlying model of RDF. The only guarantee that s/he has is the control over the repository used to make trust evaluations. Therefore, it is important for the user to understand whether this assumption is consistent with his desired meaning for a policy that uses negation, given its goals.

4.3 Limitations of Horn Clauses

The use of horn clauses allows one to express easily reasoning based on material implication, linking a fact to sets of necessary and sufficient evidences. Nonetheless, they may not be adequate to express some conditions, e.g. those based on quantity constraints and on statistical calculations.

One possible solution is to use features available in the Prolog implementation, like `findall/3` and `is/2` predicates, to overcome some limitations, especially those related to quantity restrictions. More complex conditions, like ones related to social network analysis, may be evaluated outside Prolog, given the restriction that only trusted facts are used in their evaluation. A similar approach is taken in TriQL.P [4].

5 Implementation

We implemented the idea of trusted repositories using a Java library that wraps an existing RDF repository, runs trust policies on demand and offers programmatic access to its conclusions: which facts were trusted, the facts used to justify each trusted fact and their justification levels.

To apply the trust policies, we used XSB Prolog [16]. We choose this Prolog implementation due to its ability to cope nicely with recursive predicates, avoiding infinite loops. The XSB Prolog runs the inference to find out which RDF triples are trustworthy; the Java library interfaces with the XSB Prolog code, converting the RDF triples to be analyzed to Prolog and converting the results back. These results are not added directly to the repository, although this could be easily done using reification. However, we opted for a less intrusive approach, leaving to the client application the decision to store or not the computed trust information in the repository.

Trust information is generated on demand. When new facts are added to the repository, it is necessary to run the trust engine again to update trust information. We did not develop yet a strategy for incremental update.

Justification levels are mapped to a pair (URI, number), where *URI* denotes the justification class and *number* denotes the *justification degree*, which is an integer that fulfils the following relation:

$$j_1 = (URI_1, n_1), j_2 = (URI_2, n_2), URI_1 = URI_2, n_1 \geq n_2 \rightarrow j_1 \geq j_2$$

This conveys the formal model's restriction that two justification levels are comparable when they belong to the same justification class. The use of an integer eases the task of computing and using the results of trust. However, this number just expresses ordering: there is nothing in the formal model that guarantees meaningfulness of arithmetic operations on these numbers.

RDF statements are converted to Prolog facts with the following form:

```
fact(Subject, Predicate, Object, Context, ID).
```

The values of these elements can be URIs, which are mapped to Prolog atoms, strings, which are also mapped to atoms, and numbers. Blank nodes are mapped back and forth to fake URIs. Trust policies are represented as Prolog clauses. Logical conditions translate to terms; Prolog's dynamic database is used as the knowledge base (in fact, just those terms whose head are of the form fact/5).

Right now, we have a prototype of this library that wraps Jena repositories, Sesame local repositories, Named Graphs sets [6] and DBin [17] trusted repositories.

6 Related Work

The work of Gerck [10] provides a lengthy discussion on the concept of trust as reliance on information. He contrasts this concept with other commonly used definitions for trust. He also offers a conceptual framework to reason about trust levels grounded on common sense reasoning, where trust on some information is justified if the trust agent is "convinced" of its truth. Castelfranchi et al. [7] present a cognitive model of trust, rooted in multi-agent systems domain. They also characterize trust as reliance, corroborating Gerck's model with some minor modifications.

Carroll et al. [6] proposes the named graphs extension to RDF, which adds URIs to graphs under a well-defined semantics. Among other things, this extension opens interesting possibilities to express trust policies based on provenance information, as shown in their work. These policies are used to build a set of accepted graphs, which is the set of graphs that contribute to the meaning of the entire knowledge base (in their terms, the set of named graphs); when specified by trust policies, it is really the set of trusted graphs. So, an entire named graph could be trusted or not. Our previous work [2] adopted a similar view: we worked with the set of trusted RDF triples.

The application of named graphs to the trust policies field continued with the work of Bizer et al. [4] the TriQL.P browser, which enhances the Piggy Bank browser [14] with trust policies based on the named graphs extension and on TriQL.P query language [4]. Our first work on trust modeling [1] used TriQL.P to describe and apply elementary trust policies. A fundamental difference among this work and ours is the recursive nature of our model: trust is always decided based on previously trusted statements. As we needed recursive queries to implement our trust model, we moved away from TriQL.P to XSB Prolog, in order to express recursive queries more naturally. Even so, our elementary trust policies greatly resemble this policy language and may benefit from the explanation engine built into TriQL.P browser.

As our work also uses the idea of trust measurement, it can be compared to other ones that embrace this idea when specifying trust policies. Golbeck's work [11] is one of them. It offers an approach to calculate trust on a web-based social network, grounded on the concept of trust as reliance among agents in the network. Each agent states trust on some other agents; this leads to a social network whose directed edges express trust of an agent on other. This trust attitude has degrees, which are weights of the graph edges. From this model, Golbeck proposes algorithms to infer trust among people connected only indirectly through the network. We adopt a different approach to trust measurement, as we believe that assigning numbers to trust relationships and than making computation with these numbers can be misleading, as the semantics of the result is not clear, even if it obeys some kind of intuition about how trust works in real life. We use the idea of justification levels in order to incorporate the notion of trust levels while keeping the semantics clear, e.g. forbidding direct comparison of different justification classes without some underlying theory that justifies this.

The implementation of trust engines grounded on logic is shared by works like PeerTrust [15] and SULTAN [12]; they also share the idea of collecting evidence to decide trustfulness. However, none of them is concerned with Semantic Web data.

7 Conclusions and Future Work

Our goal was to build a model to capture, represent and apply trust policies of an agent in the scenario of Semantic Web knowledge bases, while preserving real-world semantics of trust. We first specified a model capturing relevant aspects of the trust concept, such as reliance, subjectivity, dynamism, justification, measurement. Then proceeded to build a horn logic model to express trust policies that may be built incrementally through the concepts of elementary trust policies and root trust policies. We presented a motivating example where the described model offers a solution and described a test implementation we have made, which can be used as a trust layer for an RDF repository.

The next steps in this work include a deeper study of justification levels, in order to provide a more solid theoretical background to this concept; building guidelines for defining trust policies, possibly yielding a method; the explicit inclusion of non-monotonocity in the model, especially negation as failure; exploring the idea of trust delegation, i.e. using trust information from other agents. We also plan to develop a case study in a realistic scenario, such as Social Semantic Desktops, Semantic Web Browsing and Social Semantic Collaborative Filtering, with large trust policies using RDF data.

8 Acknowledgements

This research was sponsored by UOL (www.uol.com.br), through its UOL Bolsa Pesquisa program, process number 20060601215400a. We would like to thank the reviewers for their valuable comments and suggestions.

9 References

1. Almendra, V. S., Schwabe, D. Real-world Trust Policies. In Proceedings of Proceedings of the Semantic Web and Policy Workshop, 4th International Semantic Web Conference (Galway, Ireland, November 2005).
2. Almendra, V. S., Schwabe, D., Casanova, M. A. Towards Real-world Trust Policies. Technical report MCC42/05, Informatics Department, PUC-Rio (2005).
3. Bizer, C., Carroll, J. Modeling Context using Named Graphs. Semantic Web Interest Group Meeting, March 2004, Cannes, France.
4. Bizer, C., Cyganiak, R., Maresch, O., Gauss, T. TriQL.P - Trust Policies Enabled Semantic Web Browser. http://www.wiwiss.fu-berlin.de/suhl/bizer/TriQLP/browser/
5. Bonatti, P., Vimercati, S. C., Samarati, P. An Algebra for Composing Access Control Policies. ACM Transactions on Information and System Security, Vol. 5, No. 1, February 2002, Pages 1–35.
6. Carroll, J. J., Bizer, C., Hayes, P., Stickler, P. Named Graphs, Provenance and Trust. Technical report HPL-2004-57 (2004).
7. Castelfranchi, C., Falcone, R. Social Trust: A Cognitive Approach. In: Castelfranchi, C.; Yao-Hua Tan (Eds.): Trust and Deception in Virtual Societies. Springer-Verlag (2001).
8. Decker, S., and Frank, M. R. The Networked Semantic Desktop. In Proceedings of the WWW2004 Workshop on Application Design, Development and Implementation Issues in the Semantic Web (New York, NY, USA, May, 2004).
9. Decker, S., Sintek, M., Billig, A. et al. TRIPLE - an RDF Rule Language with Context and Use Cases. In Proceedings of W3C Workshop on Rule Languages for Interoperability, (Washington, DC, USA, April 2005), W3C, 27-28.
10. Gerck, E. Toward Real-World Models of Trust: Reliance on Received Information. http://www.safevote.com/papers/trustdef.htm.
11. Golbeck, J., Parsia, B., Hendler, J. Trust Networks on the Semantic Web. ISWC'03.
12. Grandison, T., Sloman, M. Trust Management Tools for Internet Applications. Proc 1st Intl. Conference on Trust Management, May 2003, Crete.
13. Guha, R., McCool, R., and Fikes, R. Contexts for the Semantic Web. In Proceedings of the ISWC'04 (Hiroshima, Japan, November 2004), Springer.
14. Huynh, D., Mazzocchi, S., and Karger, D. Piggy Bank: Experience the Semantic Web Inside Your Web Browser. In Proceedings of ISWC'05 (Galway, Ireland, November 2005), Springer.
15. Nejdl, W., Olmedilla, D., Winslett, M. PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. Workshop on Secure Data Management in a Connected World (SDM'04) in conjunction with 30th International Conference on Very Large Data Bases, Aug.-Sep. 2004, Toronto, Canada
16. Rao, P., Sagonas, K. F., Swift, T., Warren, D. S. and Freire, J. 14: A System for Efficiently Computing Well-Founded Semantics. http://citeseer.csail.mit.edu/rao97xsb.html
17. Tummarello, G., Morbidoni, C., Puliti, P., Piazza, F. The DBin Semantic Web platform: an overview. http://www.instsec.org/2005ws/papers/tummarello.pdf.

An Access Control Model for Protecting Semantic Web Resources

Sara Javanmardi, Morteza Amini and Rasool Jalili

Network Security Center, Computer Engineering Department,
Sharif University Of Technology, Tehran, Iran
{s_javanmardi, m_amini}@ce.sharif.edu, jalili@sharif.edu

Abstract. Semantic Web is a vision for future of the current Web which aims at automation, integration and reuse of data among different Web applications. Access to resources on the Semantic Web can not be controlled in a safe way unless the access decision takes into account the semantic relationships among entities in the data model under this environment. Decision making for permitting or denying access requests by assuming entities in isolation and not considering their interrelations may result in security violations. In this paper, we present a Semantic Based Access Control model (SBAC) which considers this issue in the decision making process. To facilitate the propagation of policies in these three domains, we show how different semantic interrelations can be reduced to the subsumption problem. This reduction enhances the space and time complexity of the access control mechanisms which are based on SBAC. Our evaluations of the SBAC model along with experimental results on a sample implementation of the access control system show that the proposed model is very promising.

1 Introduction

Semantic Web is an extension for the current Web which gives information a well–defined meaning, making machines capable of interpreting and processing the information. The shift from current Web to semantic aware environments such as the Semantic Web poses new security requirements [1, 2] specially in the field of access control. Access control is a mechanism that allows owners of resources to define, manage and enforce access conditions applicable to each resource [3]. A semantic aware access control mechanism should assure that *only* eligible users are authorized to be granted an access right and each eligible user must be able to access *all* the resources that s/he is authorized for [4]. Traditional access control models like MAC, DAC and RBAC fail to address these issues since they do not consider the rich semantic relations in the data model under the Semantic Web [5]. In other words, decision making based on isolated entities while ignoring the semantic interrelationships among them may result in *illegal inferences* by unauthorized users and *incomplete granting of access rights*. For an example of an illegal inference, consider a concept ‘Credit Card’ which is the union of concepts ‘Master Card’ and ‘VISA Card’. If a user is eligible to know about the latest transactions on credit cards issued by a bank while s/he is prevented from accessing the same information for VISA cards, then s/he can guess some information about them which is illegal. On the other hand, when a bank authority needs to know some information about the 'Letter of Credit' concept for some decision making then s/he should be also authorized for reading the information about an equal concept like 'Documentary Credit'.

To overcome these challenges, there is a need for semantic aware access control systems. In this paper, we present a Semantic Based Access Control model (SBAC) that authenticates users based on the credentials they offer when requesting an access right. Ontologies are used for modeling entities along with their semantic interrelations in three domains of access control, namely subject domain, object domain and action domain. Decision making in SBAC for permitting or denying an access request is automated by inference engines. We show how semantic interrelations can be used in the authorization process; and for enhancing the expressiveness of authorization rules defined in SBAC, we show how rule languages like SWRL [6] can be applied. Since a general semantic relation called *subsumption* can facilitate the policy propagation, in SBAC we try to reduce different semantic interrelations to the subsumption problem.

The remainder of this paper is as follows: Section 2 describes the related works on this topic and section 3 states the fundamentals of SBAC. Semantic authorization flow of access rights in different levels of an ontology are described in section 4. In section 5, the formal definition of SBAC is presented and it is shown how the reasoning can be done in different domains of access control. Our proposed architecture for implementing the SBAC model is presented in section 6 and the experimental results and qualitative evaluations of the model are described in section 7. Finally, section 8 underlines some conclusions and future research lines.

2 Related Works

Access control systems for protecting Web resources along with credential based approaches for authenticating users have been studied in recent years [3]. With the advent of Semantic Web, new security challenges were imposed on security systems. Bonatti et al in [2] have discussed open issues in the area of policy for Semantic Web community such as important requirements for access control policies. Developing security annotations to describe security requirements and capabilities of web services providers and requesting agents have been addressed in [7]. Fig. 1 shows the trend of developing security issues in the Semantic web.

Object-Oriented authorization models for databases were the first models that tried to consider the semantic relationships for authorization. Such models showed the effect of the semantic relationships like subclass/superclass in access decision making [8]. File–level access control systems were studied in [9] for protecting HTML resources. In the next layer, there are XML based approaches such as XACML (eXtensible Access Control Markup Language) [10] and XR-BAC (XML Role-Based Access Control) [11] that have attempted to express

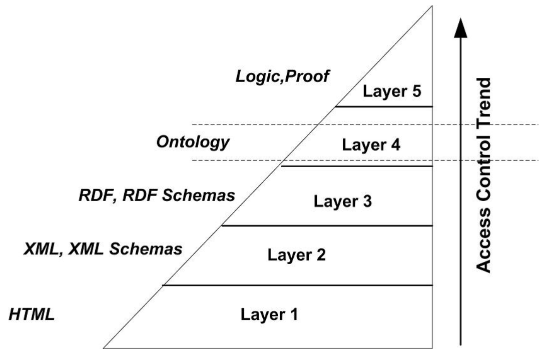


Fig. 1. SBAC in the Stack of Semantic Web

policies for controlling accesses to XML resources. Finin et al have proposed policy languages like Rei [12] based on Semantic Web languages like RDF and DAML+OIL and have developed a framework, Rein, based on Rei. In the ontology layer, Qin et. al. [4] proposed a concept level access control model which considers some semantic relationships in the level of concepts in the objects domain. In this paper, we present SBAC as an access control model based on OWL [13] ontology language that considers semantic relationships in different levels of an ontology (Concept, Property, Individual) and in all the domains of access control (Subject, Object, Action). For enhancing the expressiveness and inference abilities, SBAC uses SWRL, a Horn clause rule extension to OWL.

3 Introduction to SBAC

Fundamentally, SBAC consists of three basic components: Ontology Base, Authorization Base and Operations. Ontology Base is a set of ontologies: Subject–Ontology (SO), Object–Ontology (OO) and Action–Ontology (AO). These ontologies are described in the following:

OO : is an Object Ontology for describing objects. Objects are entities which are accessed and/or modified. An Object–Ontology shows the structure in which the objects (Concepts, Individuals and Properties) are organized along with the semantic relationships among them. Fig. 2 is an example OO. It shows a part of a Bank-Service ontology. The ovals show concepts and individuals and labels on the directed arcs show axioms and properties. Individuals are represented by ovals that have arcs with 'Is_A' labels to other ovals.

SO : is the Subject Ontology where subjects are active entities which require access to objects. Subjects are represented using concepts or individuals in a Subject-Ontology. Fig. 3.a shows a Subject-Ontology which is based on credentials. Presenting credentials determine users eligibility for accessing a resource.

AO : Actions depend on the type of the actions that subjects aim to execute on objects. Each action type is a concept in the action ontology. Fig. 3.b demonstrates an example of Action Ontology.

By modeling the access control domains using ontologies, SBAC aims at considering semantic relationships in different levels of an ontology to perform inferences to make decision about an access request. Authorization Base is a set of authorization rules in form of $(s, o, \pm a)$ in which s is an entity in SO , o is an entity defined in OO, and a is an action defined in AO. In other words, a rule determines whether a subject which presents a credential s can have the access right a on object o or not. Predefined access rights can be saved in Authorization Base in the form of authorization rules and for making decisions for incoming requests (permit/deny), inference is done based on the semantic relationships between the requested authorization and the explicit authorization rules in Authorization Base. In fact, inferences on the explicit authorization rules result in some implicit authorization rules. For example, if an explicit authorization rule states that a subject can read an object of type "Account", then if s/he requests an access right to read a subobject of type "ShortTermDeposit", then the latter can be inferred from the former without having its authorization rule explicitly. Since SBAC works based on inference, for preventing propagation of same decision (permit/deny) on all the inferred rules, it allows the definition of exception rules with higher priority. For example, an exception rule can be defined if the authority of a bank wants to prohibit the credit cards issued from a specific bank from settling money to any account in $Bank_x$ while there is another explicit authorization rule that lets all credit cards settle money in any account.

4 Semantic Authorization Inference

Different semantic relations in an ontology result in semantic authorization flow among entities in different levels of that ontology. OWL is the W3C recommendation for representing ontologies in a machine–processable format. To automate the inference process in SBAC, we used this language since its well–defined structure lets machines automatically process the knowledge described in it; besides it supports strong semantic relations among concepts. Based on OWL, we have identified three levels: concept–level, individual–level and property–level where the semantic authorization flow can occur in each level or between different levels. To simplify the effect of semantic authorization flow in decision making, first we classify the possible semantic inferences that can occur, and then we explain different types of inferences in each category. This classification is done based on the fundamental OWL structures [13] which are OWL Class Axioms, Individual Axioms, Property Characteristics and Property Restriction.

– **Concept-Concept (C-C)**: Inference can be done in the level of concepts (between two concepts) in an ontology. Concept constructors in OWL result in new concepts with an intrinsic semantic authorization flow. For example,

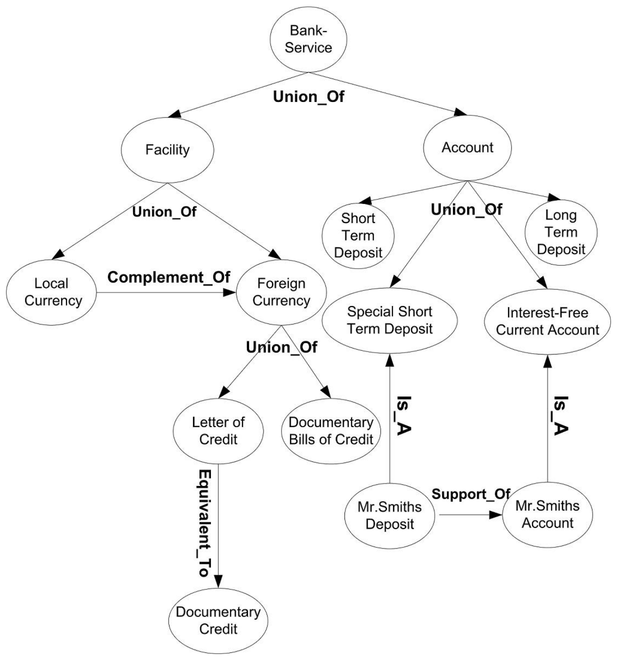


Fig. 2. A Part of Bank-Service Ontology

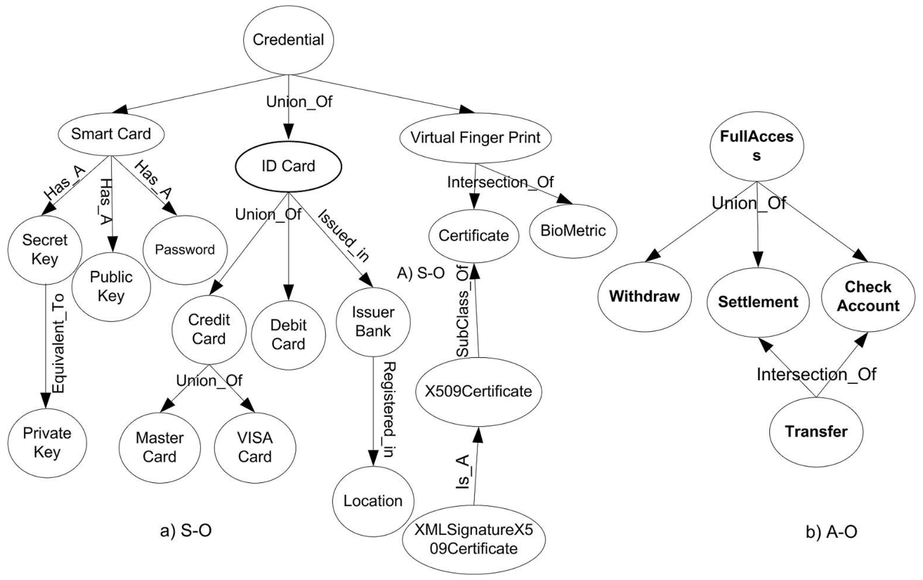


Fig. 3. a) A Credential ontology for modeling the subject domain. b) A part of executable actions on the Bank-Services ontology.

when the concept ‘Credit Card’ is defined as the union of ‘Master Card’ and ‘VISA Card’, then access rights such as eligibility of the owner of a credit card for checking an account will be propagated to both the owner of a ‘Master Card’ and the owner of a ‘VISA Card’.

– **Concept-Individual (C-I)**: All the individuals are influenced by the access conditions enforced on the concept they belong to.
– **Individual-Individual (I-I)**: Individual axioms cause this kind of authorization flow. For example the ‘same as’ axiom states that two individuals are semantically equal, hence the access conditions on each of them should be applied on the other one too.
– **Property-Concept (P-C)**: The semantic authorization flow from properties to concepts happens when an access right on a property is granted. A property is interpreted by a set of ordered pairs of individuals where the first individual is in the domain of the property and the latter is in the range of it. Therefore, any access right on a property can result in the same access right on the domain and range of the property. For example, when a subject can modify a property, s/he should be able to access the domain and range of that property.
– **Property-Property (P-P)**: Semantic relations between various properties can result in new properties which are necessary to decision making but are not explicitly mentioned in the ontology. For example, when a bank authority wants to prevent master cards supported by Asian banks from settling money in a special account by defining $(AsianMasterCards, Account_x, -settelement)$, by having knowledge on two properties ‘Issued_in’ and ‘Registered_in’, the new property of ‘Supported_by’ can be made. The related SWRL rule is as follows:

$$Registered\_in(Bank_x, Asia) \wedge Issued\_in(MasterCard, Bank_x) \\ \rightarrow Supported\_by(MasterCard, AsianBank)$$

– **Property-Individual (P-I)**: All the individuals are influenced by the access conditions enforced on the property that they belong to. Moreover, property characteristics like being transitive or symmetric imply membership of some new individuals to the same property which are also affected by the access conditions defined on the property. For example, if we define the ‘Support_Of’ property as a symmetric property then by having the knowledge that $(Account_x, Account_y)$ is an individual of a property then it can be inferred that $(Account_y, Account_x)$ is also an individual of that property. An SWRL rule like the following can be added for the inference:

$$Support\_of(Account_x, Account_y) \rightarrow Supported\_of(Account_y, Account_x)$$

– **Concept-Property (C-P)**: When an access right on a concept is granted, then there would be semantic authorization flows from this concept to the restricted concepts that are result of property restrictions on this concept. For example, when a subject is eligible to ‘Check_Balance’ of some credit cards

then s/he should be authorized to ‘Check_Balance’ of any restricted concept like $Issued\_In.Bank_x$ which returns credit cards issued in the $Bank_x$.
It is worth noting that the ontology languages in the fourth layer of the Semantic Web stack are not expressive enough to support all of the inference classifications that should be performed in the machine level. Fig. 4 shows the degree of coverage of OWL DL and SWRL. As can be seen in this figure, using SWRL rules provide better expressivity.

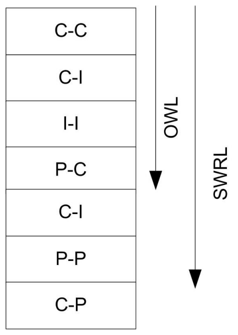


Fig. 4. Comparison of inference support in OWL and SWRL

4.1 Reduction to Subsumption

Different kinds of semantic relations and inference problems based on them motivated us to reduce the possible inferences on the semantic relationships in OWL DL to the general problem of *Subsumption*. Checking the subsumption property is the basic reasoning method of description logics [14]. Given two concepts C and D and a knowledge base Σ, the following expresses that D subsumes C in Σ: $\Sigma \models C \sqsubseteq D$. This reasoning based on subsumption proves that D (the subsumer) is more general than C (the subsumee). In SBAC, we use a variant of the subsumption relation which is represented by $\preceq$ and not only handles concepts but also considers individuals. It is defined as follows:

$$A \preceq B \; = \begin{cases} A \sqsubseteq B, & if\ A\ and\ B\ are\ concepts \\ A\ Is\_A\ B, & if\ A\ is\ an\ individual\ and\ B\ is\ a\ concept \\ A\ sameAs\ B, & if\ A\ and\ B\ are\ individuals \end{cases}$$

When there is $A \preceq B$ relation between A and B, the authorization rules enforced on B should also be enforced on A. Table 1 shows the reduction based on OWL class axioms. Table 2 is for individual axioms and Table 3 shows the reductions for OWL Property Restrictions. Table 4 shows SWRL rule definition for OWL Property Characteristics.

5 Formal Definition of Concepts in SBAC

This section presents a formal definition of the topics described informally in preceding sections. SBAC is defined by the triple $(OB, AB, Oprs)$. OB stands

Table 1. Reduction in the Scope of OWL Class Axioms

OWL Constructors	**Affected Group**	**Reduction to Subsumption**
C subClassOf D	C-C, C-I	$C \preceq D$
C equivalentClass D	C-C, C-I	$C \preceq D \wedge D \preceq C$
C disjointWith D	C-C, C-I	$C \preceq \neg D \wedge D \preceq \neg C$
C intersectionOf $C_1, \ldots, C_n$	C-C, C-I	$C \preceq C_1 \wedge \ldots \wedge C \preceq C_n$
C unionOf $C_1, \ldots, C_n$	C-C, C-I	$C_1 \preceq C \wedge \ldots \wedge C_n \preceq C$
C complementOf D	C-C, C-I	$C \preceq \neg D \wedge \neg D \preceq C$
C one of Enumeration E $\{\ldots\}$	C-C, C-I	$C \preceq E$
P1 subPropertyOf P2	C-C, C-I	$Domain(P1) \preceq Domain(P2)$ $Range(P1) \preceq Range(P2)$
P1 equivalentProperty P2	C-C, C-I	$Domain(P1) \preceq Domain(P2)$ $Range(P1) \preceq Range(P2)$ $Domain(P2) \preceq Domain(P1)$ $Range(P2) \preceq Range(P1)$

Table 2. Reduction in the Scope of OWL Individual Axioms

OWL Individual Axioms	**Affected Group**	**Reduction to Subsumption**
I1 differenetFrom I2	No Affect	–
allDifferent	No Affect	–
sameAs(I1,I2)	I-I	$I1 \preceq I2$ $I2 \preceq I1$

for Ontology Base which contains decision making ontologies (OO, SO, AO). AB stands for Authorization Base that includes explicit authorization rules. $Oprs$ are the operations that can be performed on the Authorization Base.

$SBAC = (OB, AB, Oprs)$
$OB = \{Ont \mid Ont = SO \vee Ont = OO \vee Ont = AO\}$
$Ont = (C, T, \leq_C, \leq_T, R, A, \sigma_A, \sigma_R, \leq_A, \leq_R)$
$AB = \{(s, o, \pm a) \mid s \in SO \ \wedge \ o \in OO \ \wedge \ a \in AO\}$
$Oprs = (CA, Grant, Revoke)$

In the definition of ontology (Ont), which is from [15], C is a set of concepts, $\leq_C$ is the subsumption relation between concepts. The other semantic relations are presented by $\sigma_R : R \rightarrow C \times C$. $\leq_R$ shows the hierarchy among Object Properties, meaning one property is subproperty of another property. T is a set of datatypes with a hierarchy of datatypes, $\leq_T$. DataType Properties are presented by $\sigma_A : A \rightarrow C \times T$ [13].

Access rights are stored in AB in the form of Authorization rules where:

$$AB \subseteq S \times O \times A$$

Definition (Authorization Rule)

Table 3. Reduction in the Scope of OWL Property Restriction

OWL Property Restriction	Affected Categories	Reduction to Subsumption
C allValuesFrom(P,D)	P–C, C–C, C–I	$C \preceq Domain(P)$ $D \preceq Range(P)$
C someValuesFrom(P,D)	P–C, C–C, C–I	$C \preceq Domain(P)$ $D \preceq Range(P)$
C minCardinality(P)	P–C, C–C, C–I	$C \preceq Domain(P)$
C maxCardinality(P)	P–C, C–C, C–I	$C \preceq Domain(P)$

Table 4. SWRL Rule Definition in the Scope of OWL Property Characteristics

OWL Property Characteristics	Has Effect	Affected Categories	SWRL Rules
TransitiveProperty	Yes	P–I, P–P	$P(a,b) \wedge P(b,c) \rightarrow P(a,c)$
SymmetricProperty	Yes	P–I, P–P	$P(a,b) \rightarrow P(b,a)$
FunctionalProperty	No	No Affect	$P(a,b) \wedge P(b,c) \rightarrow P(a,c)$
InverseOfProperty	Yes	P–I, P–P	$P(a,b) \rightarrow P^{-1}(b,a)$
InverseFunctionalProperty	No	No Affect	–

An authorization rule is a triple like $(s, o, \pm a)$ where $s \in SO$, $o \in OO$, and $a \in AO$.

The knowledge base consists of explicit authorization rules and is formally defined $AB \subseteq S \times O \times A$. An authorization rule is a triple $(s, o, +a)$ where $s \in SO$, $o \in OO$, $a \in AO$.

Definition (Operations)
The operations are executed on AB and are for making decision about a request, granting an access right or revoking an access right and the formal definition is $Opr = (CA, Grant, Revoke)$.

– $CA(s, o, a)$: the function of decision making is $CA : S \times O \times A \rightarrow \{true, false\}$. $CA(s, o, a) = true$, if $(s, o, +a) \in AB$ or there is an authorization rule $(s_i, o_j, a_k) \in AB$ such that $(s_i, o_j, +a_k) \rightarrow (s, o, +a)$. $CA(s, o, a) = false$, if $(s, o, -a) \in AB$ or there is an authorization rule $(s_i, o_j, a_k) \in AB$ such that $(s_i, o_j, -a_k) \rightarrow (s, o, -a)$. Otherwise, due to the close policy the function returns 'False'. The reasoning '$\rightarrow$' from (s, o, a) to (s_i, o_j, a_k) can be performed on domains subject SO, object OO or action AO. Definition of function CA is as follows:

$$CA(s,o,a) = \begin{cases} True, & (s,o,+a) \in AB \vee (\exists (s_i,o_j,+a_k) \in AB : \\ & \quad (s_i,o_j,+a_k) \rightarrow (s,o,+a)) \\ False, & otherwise \end{cases}$$

Conflicts are possible in $CA(s, o, a)$ in the time of decision making. Exception rules are one of the sources of conflicts. Since for making a decision about a request two conflicting inferences can lead to different results, conflict resolution is necessary in SBAC. Inference from exception rules should have higher priority than inference from other explicit rules. Hence for resolving the conflict, the inference from the most specific rule which is the most specific exception takes precedence than other inferences. This conflict resolution policy is possible since the conflicting sources of inference are on the same inference path and comparing the conflicting rules is possible. In the cases that the conflicting rules are not comparable or in other words they are not on the same inference path, a "negative take precedence" policy which gives the priority to the negative authorization rule is used for resolving the conflict.

– $Grant(s, o, a)$: Granting an authorization (s, o, a) means inserting the rule in AB . This operation is executed by the operation $Grant(s, o, a)$, which returns the Boolean value True if the rule is added and False if the rule can not be added to AB.

```
Grant(s,o,a):
   if $(s, o, a) \in AB$ or $CA(s, o, a) = true$ then return false
   else
        add (s,o,a)
        return True
```

– $Revoke(s, o, a)$: Revoking an authorization (s, o, a) means deleting it from AB. This operation is executed by the operation $Revoke(s, o, a)$, which returns the Boolean value True if the rule is deleted and False if the rule can not be deleted from AB.

```
Revoke(s,o,a):
   if $(s, o, a) \in AB$ then
         $delete\ (s, o, a)$
         return True
   else return false
```

5.1 Authorization Propagation

In this section, we explain how reducing the inference problem to the subsumption problem can result in an effective way for authorization propagation in three domains of access control. In the domains of subjects and objects, the authorizations are propagated from subsumee to subsumer; but the propagation of access rights in the domain of actions is different and the negative access rights will be propagated from subsumer to subsumee. It means that the subsumee can not have a positive right while the subsumer does not have it. But the positive access rights are propagated in the opposite direction. In other words, if the subsumee has a positive access right, the subsumer should also have it. The following is a formal description of the propagation mechanism:

– **Propagation in subject domain**: Given $(s_i, o, \pm a)$, If $s_j \preceq s_i$ then the new authorization rule $(s_j, o, \pm a)$ can be derived by inference from s_i to s_j, we denote this rule as $(s_i, o, \pm a) \rightarrow (s_j, o, \pm a)$.

- **Propagation in object domain**:Given $(s, o_i, \pm a)$, If $o_j \preceq o_i$ then the new authorization rule $(s, o_j, \pm a)$ can be derived by inference from o_i to o_j, we denote this rule as $(s, o_i, \pm a) \rightarrow (s, o_j, \pm a)$.
- **Propagation in action domain**:
  - Given $(s, o, +a_i)$, If $a_j \preceq a_i$ then the new authorization rule $(s, o, +a_i)$ can be derived by inference from a_i to a_j, we denote this rule as $(s, o, +a_i) \rightarrow (s, o, +a_j)$.
  - Given $(s, o, -a_j)$, If $a_j \preceq a_i$ then the new authorization rule $(s, o, -a_i)$ can be derived by inference from a_j to a_i, we denote this rule as $(s, o, -a_j) \rightarrow (s, o, -a_i)$.

6 A Proposed Architecture for implementing the SBAC Model

Fig. 5 shows our proposed architecture for implementing the SBAC model. This architecture shows the details of the authorization process which is used during the decision making process in SBAC. This architecture contains a number of external components and a number of authorization components which are described in the following:

External Components: External components are subjects, ontological definitions of credentials, objects, and actions, Reputation system, and administration tools. Subjects are the ones that request for access rights. ontological definitions of credentials, objects, and actions are as described in previous sections. The reputation system is used for checking the validity of credentials that are provided by subjects. Administration tools are used for managing the Authorization Base. For example, adding or revoking rules in this base are performed using these tools.

Authorization Components: Authorization components are as follows:

- Authorization Base: which includes the explicit authorization rules that are defined by security administrators of system.
- Ontology Base: which includes ontologies that describe different domains of access control.
- Ontology Parser: which receives an ontology as input and applies the reduction algorithm of section 4.1 on it.
- Reduced Ontologies: these are the ontologies that are parsed by the Ontology Parser component and are ready to be used with the Semantic Authorizer component.
- Semantic Authorizer: which after receiving a request from a subject uses its inference engine to determine whether this subject should be authorized to access the requested object.

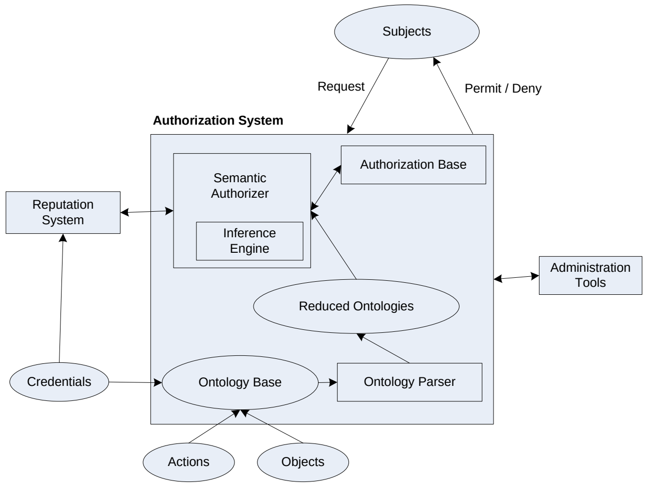


Fig. 5. Proposed Architecture for implementing the SBAC model

7 Evaluation

The most obvious advantage of SBAC compared with other access control models is its Semantic–awareness property. But, besides *Semantic-awareness* SBAC has the following advantages:

- **Interoperability**: Interoperating across administrative boundaries is achieved through exchanging authorizations for distributing and assembling authorization rules. The ontological modeling of authorization rules in SBAC results in a higher degree of interoperability compared with other approaches to access control. This is because of the nature of ontologies in providing semantic interoperability.
- **Expressivity**: The expressiveness of the security policies directly depends on the expressiveness of the language using which the policies are described. SBAC authorization rules are defined using OWL DL which is based on an expressive description logic namely, $\mathcal{SHOIN}(D)$ [16]. For enhancing the expressiveness, SBAC also uses SWRL rules.
- **Ease of Implementation and Integration with Semantic Web technologies**: Security models designed for Semantic Web should be compatible with the technology infrastructure under it. In other words, the implementation of security mechanisms should be possible based on the semantic expression models. SBAC is designed based on the widely accepted semantic web languages, OWL and SWRL, therefore its implementation can be easily achieved by existing tools designed for working with these languages.
- **Generality**: Modeling different domains of access control has added a considerable generality to the model. In the subject domain, SBAC uses credentials which are going to be universally used for user authentication. In

the domain of object, different kinds of resources such as web pages or web services can be modeled and can be identified by their URI in authorization rules.

- **Space Efficiency**: Implicit authorization in SBAC results in a certain level of efficiency since it is not necessary to store all the authorizations rules explicitly when they can be inferred from other stored authorizations. Besides implicit authorizations allow continuous changing of semantic relations (ontology evolution).
  On the other hand, as is shown in Fig. 6, for representing the expression $C = C_1 \cup \ldots \cup C_n$ using RDF triples, $2n + 1$ triples are required. While as is shown in Fig. 7 after reducing this expression using the subsumption relation, only n triples are required. This situation is valid for most of the other OWL constructors. In order to experimentally show this fact, we generated random ontologies and created a program called *OntGenerator* which receives three parameters, namely *conceptCount*, *expCount*, and *expMaxSize*, as input parameters and generates a random ontology based on the values of these parameters. conceptCount shows the number of atomic concepts and expCount shows the number of complex concepts in this ontology. expMaxSize shows the maximum number of concepts (whether atomic or complex) that are used for creating each complex concept.
  Table 5 shows number of statements in standard and reduced ontologies for random ontologies generated for different values of conceptCount, expCount, expMaxSize. As can be seen in this table, the number of statements is reduced after applying the reduction algorithm on these ontologies. This shows that, SBAC needs to work with smaller ontologies and therefore it requires a lower space capacity.

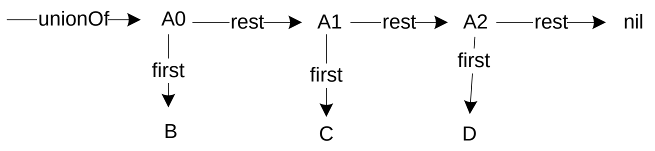


Fig. 6. Representing $A = B \cup C \cup D$ using RDF triples

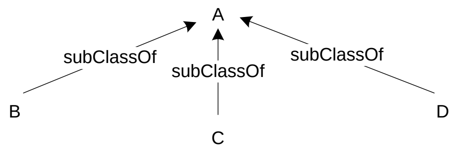


Fig. 7. Reduced version of $A = B \cup C \cup D$

- **Low Response Time**: most of the time complexity of decision making functions refers to the reasoning part. Since we have reduced reasoning problems to the subsumption problem and because of the existence of highly efficient subsumption reasoners, the response time of SBAC is very promising. For evaluating the reasoning time of SBAC, we designed an experiment. In our experiment, we used the PELLET reasoner which is a highly efficient OWL

Table 5. Number of statements in standard and reduced ontologies

conceptCount	expCount	expMaxSize	Statements of Standard Ontology	Statements of Reduced Ontology
100	20	10	390	239
1000	20	10	1262	1130
1000	100	10	2382	1688
500	200	10	3158	1825
1000	200	10	3600	2300
5000	500	20	16194	10593

Table 6. Comparison of reasoning times

conceptCount	expCount	expMaxSize	Reasoning time on Standard Ontology	Reasoning time on Reduced Ontology
100	20	10	969	843
1000	20	10	7484	1156
1000	100	10	7907	1172
500	200	10	3938	1141
1000	200	10	8781	1219
5000	500	20	156687	2204

DL reasoner for reasoning on standard ontologies. On the other hand, since reduced ontologies only include subsumption relation between concepts, we designed and implemented a fast reasoning engine which can only handle the subsumption relation but in a better time period compared with reasoners such as PELLET. In fact, this point that SBAC can do its decision making using reasoning engines that only need to handle the subsumption relation is one of the biggest strengths of this model. Table 6 shows a comparison of reasoning time of the PELLET reasoner which must work with the standard ontology and the reasoning time of our reasoner which can work with the reduced ontology. As can be seen in this table, our reasoner can do the decision making process in a smaller time period.

8 Conclusions and Future Work

In this paper, we presented SBAC as an access control model for protecting Semantic Web resources. SBAC takes into account semantic interrelations among entities in the domains of decision making of access control. Automated decision making in SBAC for permitting or denying an access request is done through inference processes based on the semantic relation among entities. We have shown that SBAC can provide space-efficient expression of rules with faster reasoning time than by using a standard ontology.

One of the useful features that is not addressed in SBAC is context-awareness. For example, currently a security administrator can not specify "(s,o,a) allowed only between 9am–5pm". One of our future works is to extend SBAC to DSBAC

(Dynamic SBAC) which uses context ontologies to capture the current context and use it for more expressive reasoning.

To enhance the expressiveness of the model for describing the authorization rules, more expressive logics in logic layer of Semantic Web stack can be applied. Since more expressive logics are less decidable, approaches like client based access control approaches [17] seems suitable for delegating some access control phases to the client side.

References

1. Hengartner, U., Steenkiste, P.: Exploiting information relationships for access control. In: proceeding of third IEEE International Conference on Pervasive Computing and Communications, Percom 2005, Kauai, Island HI (2005) 278–296
2. Bonatti, P.A., Duma, C., Fuchs, N., Nejdi, W., Olmedila, D., Peer, J., Shahmehri, N.: Semantic web policies – a discussion of requirements and research issues. In: ESWC 2006. (2006) 712–724
3. Samarati, P., di Vimecati, S.C.: Access control: Policies, models, architectures. In: FOSAD 2000. Volume 2171 of LNCS., Springer-Verlag (2001) 137–196
4. Qin, L., Atluri, V.: Concept-level access control for the semantic web. In: ACM Workshop on XML Security, Fairfax, VA, USA (2003) 94–103
5. Yague, M., Mana, A., Lopez, J.: Applying the semantic web layers to access control. In: Proceeding of 14th IEEE International Workshop on Database and Expert Systems Applications. (2003) 622–626
6. Hayes, P., Horrocks, I., Patel-Schneider, P., Boley, Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML (2004)
7. Denker, G., Kagal, L., Finin, T., Paolucci, M., Sycara, K.: Security for daml web services: Annotation and matchmaking. In: Proceedings of the 2nd International Semantic Web Conference, Sanibel Island, Florida, USA (2003)
8. Rabitti, F., Bertino, E., Kim, W., Woelk, D.: A model of authorization for next-generation database systems. ACM TODS **16**(1) (1991)
9. Prud'hommeaux, E.: W3C ACL System (2001)
10. Moses, T.: (eXtensible Access Control Markup Language (XACML), version 2.0)
11. Joshi, J.: Access-control language for multi domain environments. IEEE Internet Computing **8**(6) (2004) 40–50
12. Kagal, L., Finin, T., Joshi, A.: A policy language for a pervasive computing environment. In: Proceeding of 4th IEEE International Workshop on Policies for Distributed Systems and Networks. (2003) 63–74
13. Patel-Schneider, P., Hayes, P., Horrocks, I.: OWL: Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation (2004)
14. Horrocks, I.: The fact system. In: Automated Reasoning with Analytic Tableaux and RelatedMethods: International Conference Tableaux'98, Springer-Verlag (1998) 307–312
15. Ehrig, M., Haase, P., Stojanovic, N., Hefke, M.: Similarity for ontologies - a comprehensive framework. In: Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability, PAKM 2004. (2004)
16. Parsia, B., Sirin, E.: Pellet: An OWL DL Reasoner. In Moller, R., Haaslev, V., eds.: Proceedings of the International Workshop on Description Logics (DL2004). (2004)
17. Bauer, L., Schneider, M., Felten, E.: A general and flexible access-control system for the web. In: Proceedings of the 11th USENIX Security Symposium. (2002)

Context-aware Trustworthiness Evaluation with Indirect Knowledge

Santtu Toivonen[1], Gabriele Lenzini[2], and Ilkka Uusitalo[3]

[1] VTT Technical Research Centre of Finland
P.O.Box 1000, FIN-02044 VTT, Finland
santtu.toivonen@vtt.fi
[2] Telematica Instituut
P.O.Box 589, 7500 AN Enschede, The Netherlands
gabriele.lenzini@telin.nl
[3] VTT Technical Research Centre of Finland
P.O.Box 1100, FIN-90571 Oulu, Finland
ilkka.uusitalo@vtt.fi

Abstract. Commonly, when a Trustor evaluates a Trustee's trustworthiness, it is assumed that the evaluation is based on information directly available to the Trustor. This can concern for example the reputation and recommendations characterizing the Trustee. In cases of context-aware trust, this is further restricted by concentrating mainly on information in a similar enough context as is effective at trust evaluation time. However, this information is not necessarily available to the Trustor. Surprisingly, in such scenarios the literature suggests either to wait for someone else to collect the needed experience, or to trust blindly. In this paper, we discuss solutions that help the Trustor to conduct its evaluation even if direct knowledge about the Trustee is lacking. We approach this by allowing the Trustor to make use of networks connecting the Trustor and the Trustee, as well as the context information characterizing the entities appearing in these networks.

1 Introduction

Trust is an increasingly important phenomenon to grasp and support in open environments, such as the Internet, where participants are not necessarily in direct contact with each other. A common scenario is that a the subject of trust (Trustor) is searching for a service or a product (Trustee) for a certain purpose. Semi-automatic trustworthiness evaluation is of special relevance on the Semantic Web, where Trustors can be software agents in addition to human beings, and Trustees are software agents or web pages carrying information for Trustors to depend on. To perform an appropriate evaluation, Trustors request Trustees' credentials, often expressed in terms of profiles, reputation descriptions, and recommendations (cf. [1]). The difference between reputation and recommendation is that reputation is based on the Trustor's personal experiences, whereas recommendations are communicated experiences of others.

Context-awareness is also an emerging computer science trend, which takes situational details into account. Generally, in computer science context refers to any information characterizing the situation of any entities considered relevant to the interaction between a user and an application, including the user and the application themselves, as well as their surroundings [2, 3]. Note that since we are operating in environments where the entities are often software programs, it is relevant to consider their context too [4]. In the scope of the Semantic Web, one important task where the notion of context can assist is aggregation, that is, the activity of integrating data or information from multiple sources [5]. In our work aggregation is not so much directed to the semantics of descriptions characterizing various entities, but rather to combining the trustworthiness values of these entities.

Many research efforts in addition to ours also acknowledge that context information may help to define trust credentials (cf. [6, 7, 8]). In [9], we discuss context-aware trust functions; as relevant credentials, we identified the quality attributes of a Trustee, the context attributes (of the Trustee, Trustor, and the surrounding environment), the Trustee's reputation in the eye of the Trustor, as well as recommendations about the Trustee put forward by others.

Trust management frameworks operate under the assumption that the Trustor can directly access the information he requires to complete the trustworthiness evaluation [10]. In the global computing paradigm this assumption seems sometimes too optimistic. Trustee's credentials may not be available (e.g., when a new service is deployed, or when this information is protected by privacy policies), or reputation data and recommendations may refer to Trustee's behavior in contexts which are too different from the present one for them to be of use.

In this paper, we study context-aware trust establishment by considering scenarios where direct information about the Trustee is not necessarily available to the Trustor. We claim that even in such situations there are better options for Trustors to choose from than to trust/distrust blindly. For example, the Trustor can evaluate the trustworthiness of another entity somehow related to the Trustee. In many real situations humans act like this. We trust a car manufactured in a certain country, if our previous experiences with cars manufactured in that particular country are good, even if we have no experiences of that particular make. In many cases this kind of indirect evaluation suffices to accomplish a fair judgment to start with.

The particular cases we consider are the following : (i) Trustee's behavior *across contexts* is unknown to the Trustor, meaning that the Trustor has no previous knowledge of any behavior of the Trustee; (ii) Trustee's behavior in the *current context* is unknown to the Trustor, meaning that the Trustor might know the Trustee, but not how the Trustee behaves in the current context; (iii) Trustee's *recommender* and/or *recommendations* are unknown or unaccessible to the Trustor. Cases (i) and (ii) are targeted to reputation information, as they are dependent on the Trustor's knowledge and opinions on past states-of-affairs. Case (iii) relies on recommendations available to the Trustor, although the mechanisms to be considered in terms of (i) and (ii) could be plugged in it too. Note that we consider the context to be fully observable [11] to the Trustor, meaning that there is access to all relevant contextual information characterizing the Trustee, the environment, as well as the Trustor. In addition, we assume that the Trustee's quality attributes are also available to the Trustor, meaning that we do not tackle the problem of indirect quality attribute information, albeit it could follow the same lines of investigation.

The rest of the paper is organized as follows. In Section 2, we present some relevant related work. In order to pinpoint the contribution of this paper, in Section 3 we then present the baseline case where there is complete and direct information influencing trustworthiness evaluation available. We also formalize operational semantics for the trustworthiness evaluation process; it will help us later on to discuss the changes in the trustworthiness evaluation process when only indirect information is available. In Section 4, we delve into the scenarios where the Trustor has little or no reputation knowledge about the Trustee. Section 5 considers the case where recommender is not known to the Trustor. Finally, Section 6 concludes the paper and outlines some future work.

2 Related Work

The interaction between trust and context has attracted the attention of researchers only recently, and from different perspectives. In the Web Services domain, for example, context is used to anonymize the authentication procedure [12], or to decide whether granting the access to distributed resources [13, 4]. Here, differently from our approach, context is not used to evaluate the degree of users' trustworthiness. Instead, users' credentials are assumed to originate from trusted certification authorities and, together with the context, it is checked to satisfy the access conditions.

In [14], the authors use context in conjunction with content to label Semantic Web data. Only trustful (vs. merely known or untrustful) data satisfies the user-defined trust policies and is recognized by web consumers. We do not discern between trusted and merely known data in an a-priori fashion, but instead rely on recommendations and reputations to smooth out the negative effect of potentially malicious information in the evaluation process.

The problem of inferring trust from recommendations has appeared in the literature for a long time. Yahalom *et al.* [15] were one of the first to separate direct trust from recommendation-based trust and to propose an algorithm to derive new trust values given a graph of trust relationships. In [16], Beth *et al.* quantify trust, both direct and recommendation-based, as probability of the Trustee to behave as expected, and as a degree of similarity between Trustor's and recommenders' respective experiences with the Trustee. Subsequent solutions are, synthetically, extensions of the previous approaches. For example, Subjective Logic's (SL) opinions are used to model the degree of trust as well as the degree of distrust and uncertainty [17]. Alternatively, SL can be used to aggregate trust across different recommendation paths and to concatenate trust along recommendation chains [18].

Richardson *et al.* explicitly address belief composition in the Semantic Web domain [19]. They suggest software agents to maintain a table where to store their friends' beliefs as a group of statements (directed to Semantic Web data) and the agents' personal trust in their friends. The belief in unknown statements is derived though iterative merging of beliefs along paths of trust. In that work, differently from ours, there is no distinction between trust on an entity's opinion (direct trust) and trust on an entity in recommending someone else's opinion (recommendation, or referral, trust). Also, the notion of context is not visible in that work.

O'Hara *et al.* analyze costs and benefits in different paradigms (optimism, pessimism, centralized, investigation, and transitivity) of dealing with trust in Semantic Web [20]. They also identify the challenges that have motivated our research. First of all, trust must be subjective and distributed, and it also needs to be combined with personal experiences of agents. Secondly, trust should approached as context-dependent, and it needs a bootstrap procedure when there are not enough transactions to make firm judgments. Our proposal of using indirect information is an attempted answer to the bootstrap problem. It must be emphasized that existing approaches to trust management are able to deal with incomplete knowledge and uncertainty (cf. [21, 22]), but they resort mainly on the existence of recommendations. This would be impossible in case of a completely new Trustee, for example. In this paper, we argue that a Trustor can benefit from indirect sources to bring the trustworthiness evaluation to a start, and we propose methods for doing it.

3 Baseline: Direct Information Available to the Trustor

This section summarizes the formal definitions of context-aware trust evaluation functions we introduced in [9]. Additionally, it introduces and discusses an abstract operational semantics for the trustworthiness evaluation process. The operational semantics show the dynamics of the trustworthiness evaluation process when the Trustor has direct access to information characterizing the Trustee. Sections 4 and 5, which capture the main contribution of this paper, will show how this dynamics changes in reaction to using indirect knowledge.

3.1 From Context-independent to Context-aware Trust Evaluation Functions

In [9] we formalized a *context-independent* trust evaluation function as follows:

$$\texttt{trust}_{A,\sigma} : \texttt{Quality} \times \texttt{TValues} \times 2^{\texttt{TValues}} \rightarrow \texttt{TValues} \tag{1}$$

Here, $\texttt{trust}_{A,\sigma}$ is A's subjective function that returns a measure $m \in \texttt{TValues}$ of A's trust in a Trustee. The trust purpose σ (cf. [23]) indicates for what target A should trust the Trustee e.g., performing a certain task. TValues can be a set of binary values (e.g., trusted, not trusted), or discrete (e.g., strong trust, weak trust, weak distrust, strong distrust), or continuous in some form (e.g., measure of a probability or a belief). The special symbol $\perp$ represents an undefined trust measure. In all the examples of this paper we will assume TValues to be the so called "triangle of opinion" [24]; thus, a trust value is a triple $(b, d, u) \in [0, 1]^3$, and it represents the Trustor's subjective belief, disbelief and uncertainty respectively (with $b + d + u = 1$) in the Trustee to be trustworthiness for the purpose σ.

Function (1) inputs a description of the Trustee in terms of the following parameters: (a) a set $Q \in \texttt{Quality}$ of Trustee's quality attributes; (b) a trust value $m \in \texttt{TValues}$; (c) a set $M \subseteq \texttt{TValues}$ of trust values. Set Q models any information that A knows directly about Trustee, such as the Trustee's profile. Value m models the Trustee's reputation in the viewpoint of A, that is a trust value stored in A's local space. Set M

represents recommendations, which are Trustee's trust values based on the viewpoints of recommenders.

It is recognized that trust changes over time [25]. If we assume a discrete time-line, A's trust at time $i+1$ can differ from A's trust at time i. With $\texttt{trust}^i_A(B)$ we represent the trust that A has in B at time $i \geq 0$. It results from calling (1) on the inputs available to A at time i.

$$\begin{aligned}\texttt{trust}^0_{A,\sigma}(B) &:= \texttt{trust}_{A,\sigma}(Q^0_B, \bot, M^0_B) \\ \texttt{trust}^i_{A,\sigma}(B) &:= \texttt{trust}_{A,\sigma}(Q^i_B, m^i_B, M^i_B)\end{aligned} \tag{2}$$

where $Q^i_B \in \texttt{Quality}$ are the quality attributes of B at time i, $\bot$ is an undefined trust measure, $m^i_B \in \texttt{TValues}$ is the reputation of B (recommendation in A's viewpoint) at time i, and $M^i_B \subseteq \texttt{TValues}$ are recommendations on B at time i.

Definitions (1) and (2) can be extended to deal with context. Their *context-aware* counterpart is written as follows [9]:

$$\texttt{ctrust}_{A,\sigma} : \texttt{Quality} \times \texttt{Context} \times \texttt{TValues} \times 2^{\texttt{TValues}} \rightarrow \texttt{TValues} \tag{1$'$}$$

Here, `Context` models the set of context attributes, which can concern the Trustor, the Trustee, and of their interaction. An empty context is denoted with ϵ. Following the notation used for context-independent trust, with $\texttt{ctrust}^i_A(B)$ we represent the result of (1$'$) called on the inputs, among which the context $\mathcal{C}^i_{AB}$, available to A at time i. This is plugged in the context-independent trust evaluation as follows:

$$\texttt{ctrust}^i_A(B) := \mathcal{C}^i_{AB} \odot \texttt{trust}^i_A(B) \qquad i \geq 0 \tag{2$'$}$$

The operator $\odot$, such that $\epsilon \odot m = m$, returns a context-aware measure of trust, given a context-independent trust value m and a context. In [9], where we assumed $\texttt{TValues} = [0,1]$, the operator $\odot$ updates the current trust value by processing each contextual attributes in sequence. The amount of update depends on the weighting that the attributes have in Trustor's viewpoint.

3.2 Inference Rules for Context-aware Trustworthiness Evaluation

Definitions (2) and (2$'$) describe only partially the evolution of the trustworthiness evaluation process. Its understanding requires an operational formalization, that we now give in terms of an inference system. Each step of evaluation is described by an inference rule with the premises and the conclusion as predicates in the form:

$$A \xrightarrow[\mathcal{C}]{*;(i,m)}{}_\sigma B$$

stating that, for the trust purpose[4] σ, A has m degree of context-dependent $*$-trust on B, when context is $\mathcal{C}$ and time is i. Here, "$*$" stands for a class of trust. For example, we distinguish between two classes of trust relation: *functional* trust and *referral* trust [17].

[4] In the following we assume trust always implicitly referring to the same trust purpose σ, and we omit the subscript σ to make the notation more readable.

The former concerns A's trust in B performing a task; the latter concerns A's trust in B giving a recommendation about someone else doing a task. Functional trust can easily be reformulated in a context-dependent manner if it concerns A's trust in B performing a task (trust purpose) in a certain context $\mathcal{C}$. Referral trust, instead, is left context-independent: A's trust in B as a recommender does not depend on any context attributes. Naturally, this restriction could be relaxed too by letting the recommenders' contexts have influence on the trustworthiness evaluation. The predicates expressing context-dependent functional trust and referral trust, respectively, are as follows:

$$A \circ\!\xrightarrow[\mathcal{C}]{(i,m)} B \qquad A \xrightarrow{rt;(i,m)} B \tag{3}$$

Martinelli [26] adopts a similar notation for modeling functional and referral trust, but without any reference to time or context. We also identify two sub-relations of context-aware functional trust: *direct* and *indirect* trust (also pointed out in [17]). Direct trust emerges when the Trustor's trust is based on at least some personal experiences, that is, quality attributes and reputation; indirect trust is established when the Trustor judgement is based on someone else's opinions only (i.e., recommendations). We write the predicates expressing direct and indirect functional trust, respectively, as follows:

$$A \circ\!\xrightarrow[\mathcal{C}]{dt;(i,m)} B \qquad A \circ\!\xrightarrow[\mathcal{R},\mathcal{C}]{it;(i,m)} B \tag{4}$$

Here, $\mathcal{R}$ is the set of recommenders whose opinion has been considered when composing m. The semantics of context-aware trust evaluation is defined as an inference system, as depicted in Figure 1. We now comment each rule separately.

Rule (5) defines the scheme of our inference system's axioms. If A's subjective evaluation of B's qualities at time i evaluates to m and if $\mathcal{C}$ is the context available at time i, then A trusts B in measure $m' = \mathcal{C} \odot m$, where the operator $\odot$ is that of equation (2$'$). Premises in brackets (e.g., $[\texttt{trust}_A(Q^i_B)] = m$) are evaluated at a meta level.

Rules (6) formalize the operational management of recommendations. In particular, rule (6.a) shows that an indirect trust on B derives from A's referral trust in D and from the (direct) trust that D already has in B; rule (6.b) and rule (6.c) show how to concatenate referral trust along a chain of reference and how to aggregate indirect trust across multiple paths of recommendations, respectively. Accordingly to [17], rules (6.a)-(6.c) show that indirect trust always originates from a direct trust at the end of a chain of references. Referral trust can be computed as stated in [23]; we do not give the specification here. In Section 5 we will show how (6) can be applied in case the Trustor does not have a measure of referral trust in the available recommenders. Finally, rule (6.d) formalizes our proposal of dealing with context in recommendations. Context acts as a filter in favor of those recommendations experienced in contexts that are $\equiv$-related with the present context $\mathcal{C}$.

Note 1. The semantics of rules (6) are incomplete unless we give the semantics of the two operators $\otimes$ and $\oplus$.

Reasonably, $\oplus$ must be at least associative and commutative (to be order-independent) and $\otimes$ at least associative (along a chain of recommendations). Some authors (e.g.,

[27]) suggest the use of semirings [28] to deal with a network of recommendations. Alternative solutions are described in [18]. Throughout the paper we assume trust values to be Subjective Logic's opinions, and $\oplus$ and $\otimes$ to be operators on opinions called Bayesian consensus and discounting, respectively [24]. Given the opinions m, m', ω, the opinion $m \oplus m'$ reflects m and m' in a fair and equal way, whilst $\omega \otimes m$ is the opinion expressing once applied the discount rate w to m.

Note 2. Relation $\equiv \subseteq \texttt{Context} \times \texttt{Context}$ needs to be instantiated to complete the semantics of rules (6) and (7).

In its simplest form, $\equiv$ interprets as identity: a reputation or recommendation is adequate only if performed in the same context. Alternatively, $\equiv$ can be an equivalence relation between contexts—only experience performed within an equivalent context can contribute to present trust—or $\equiv$ can be a reflexive and symmetric relation modeling a semantic closeness. For example, if d is a distance between contexts, $\equiv$ can be $d(\mathcal{C}, \mathcal{C}') \leq r$, where r is the radius of the neighborhood. In case $\equiv$ is not the identity, it is reasonable to expect the derived trust to be $< m \oplus m'$. Closer study of this modified version of the rule is left as future work.

Rules (7) define how to obtain direct functional trust. More specifically, rule (7.a) models the aggregation of a direct functional trust. Rule (7.b) models our approach of dealing with reputation as a (direct) past experience that is combined with the present direct trust. Similarly to the recommendation rules, here context acts as a filter in favor of those experiences occurred in a $\equiv$-related context. Finally, rule (7.c) states that a past experience can be used as if it was a new experience presently, at the price of some trust decay (here represented by the constant discount ω).

Note 3. In rules (7.b) and (7.c) constraints over time can guide the search strategy in the past. Each strategy reflects a different attitude in considering reputation (e.g., choosing a maximal j implies the consideration of most recent experience stored in the reputation base).

Rules (8) define functional trust (the goal of our proof system) as a generalization of direct and indirect trust.

As a final remark, we observe that our inference systems allows different proof searches with different result for the same goal. Various implementations and optimization strategies are possible, but we do not discuss them in this paper.

4 Indirect Reputation Information

So far we have implicitly assumed that the Trustee's quality and contextual attributes needed in order to evaluate trust are directly available to the Trustor. In real situations, we may be obliged to relax this assumption. Consider, for example, a situation where we would like to evaluate the quality of a new scientific conference. Due to its newness, the conference is not ranked yet. Moreover, we will not find anyone known to us recommending it either. In such a situation, we basically have only two alternatives: to give up the evaluation (i.e., blindly trust/distrust), or to look for and rely on indirect information. For example, we can evaluate the prestige of the publisher of the conference

(INIT-RULES)

$$\frac{[\mathtt{trust}_A(Q^i_B) = m] \qquad [C^i_{AB} = \mathcal{C}]}{A \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i,C\odot m)} B} \tag{5}$$

(RECOMMENDATION-RULES)

$$(a)\ \frac{A \xrightarrow{rt;(i,m)} D \quad D \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i-1,m')} B}{A \circ\!\!\xrightarrow[\{D\},\mathcal{C}]{it;(i,m\otimes m')} B}\ {\scriptstyle i>0} \qquad (b)\ \frac{A \xrightarrow{rt;(i,m)} D \quad D \xrightarrow{rt;(i,m')} B}{A \xrightarrow{rt;(i,m\otimes m')} B} \tag{6}$$

$$(c)\ \frac{A \circ\!\!\xrightarrow[\mathcal{R},\mathcal{C}]{it;(i,m)} B \quad A \circ\!\!\xrightarrow[\mathcal{R}',\mathcal{C}]{it;(i,m')} B}{A \circ\!\!\xrightarrow[\mathcal{R}\cup\mathcal{R}',\mathcal{C}]{it;(i,m\oplus m')} B} \qquad (d)\ \frac{A \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i,m)} B \quad A \circ\!\!\xrightarrow[\mathcal{R},\mathcal{C}']{it;(i,m')} B \quad [\mathcal{C}' \equiv \mathcal{C}]}{A \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i,m\oplus m')} B}$$

(REPUTATION-RULES)

$$(a)\ \frac{A \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i,m)} B \quad A \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i,m')} B}{A \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i,m\oplus m')} B} \tag{7}$$

$$(b)\ \frac{A \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i,m)} B \quad A \circ\!\!\xrightarrow[\mathcal{C}']{dt;(j,m')} B \quad [\mathcal{C}' \equiv \mathcal{C}]}{A \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i,m\oplus m')} B}\ {\scriptstyle j<i} \qquad (c)\ \frac{A \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i-1,m)} B}{A \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i,\omega\otimes m)} B}\ {\scriptstyle i>0}$$

(ADDITIONAL-RULES)

$$(a)\ \frac{A \circ\!\!\xrightarrow[\mathcal{C}]{dt;(i,m)} B}{A \circ\!\!\xrightarrow[\mathcal{C}]{(i,m)} B} \qquad (b)\ \frac{A \circ\!\!\xrightarrow[\mathcal{R},\mathcal{C}]{it;(i,m)} B}{A \circ\!\!\xrightarrow[\mathcal{C}]{(i,m)} B} \tag{8}$$

Fig. 1. Abstract inference systems for context-aware trust evaluation.

proceedings, or we can look for the reputation of its program chairs and committees. In the case of a new workshop colocated with a conference having a history, we can also consider the quality of the conference when evaluating the workshop. This section studies how trust can be evaluated in such situations.

4.1 Absent Reputation Information Across Contexts

If we come across a Trustee not known to us, that is, we possess no prior reputation information about the Trustee, how should we go about evaluating the trustworthiness? One well-known solution in the literature is to ask for recommendations. In Section 5 we discuss recommendations and how to deal with them. Here, instead, we analyze a complementary solution, namely utilizing direct information of entities known to the Trustor and "related" to the Trustee (see Figure 2 (a)). Let us consider again the example about evaluating the trustworthiness of a new scientific conference. Due to the absence of any information about the conference, we can find it satisfactory to evaluate the trustworthiness of the conference proceedings publisher, as well as those of the program chairs and committee members.

From a formal point of view, the previous solution is expressed by the following additional (to the INIT-RULE) inference rule (9) where a trust relationship with B in a certain context $\mathcal{C}$ is deduced by a trust relationship with another Trustee "related" to B in the same context.

$$\frac{A \circ\!\xrightarrow[\mathcal{C}]{dt;(i,m)} D \qquad [D \sim B]}{A \circ\!\xrightarrow[\mathcal{C}]{dt;(i,m')} B} \quad m' \leq m \tag{9}$$

Here, the semantics of the rule requires us to instantiate the relation $\sim$; it can be an equivalence relation, or a reflexive and symmetric relation among Trustees that defines the concept of entity neighborhood. For example, Figure 2 (a) suggests the use of a measure of closeness among entities (see also Section 5). In this case $A \sim B$ if and only if $\texttt{cls}(A, B) \geq th$ where th is a threshold. In the following, we assume the closeness metric ranging in $[0, 1]$ where 1 stands for maximal closeness. In (9) we constrained m' to be at most m; more solutions are possible, so we left the way to calculate it unspecified. Reasonably m' depends on m and on the nature of the relationship between D and B. For example, $m' = \omega_s \otimes m$ where the opinion $\omega_s = (s, 1-s, 0)$ is the discount that reflects the closeness $s = \texttt{cls}(D, B)$ between D and B.

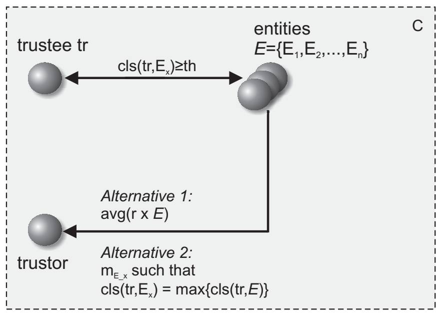
Figure 2(a) suggests also a generalization of rule (9); it considers a set of entities from which to extrapolate a measurement of B's trustworthiness. Formally the rule can be expressed as follows:

$$\frac{\bigcup_{k=1}^{N}\{A \circ\!\xrightarrow[\mathcal{C}]{dt;(i,m_k)} D_k\} \qquad [D_k \sim B]}{A \circ\!\xrightarrow[\mathcal{C}]{dt;(i,m')} B} \quad m' \leq m \tag{9'}$$

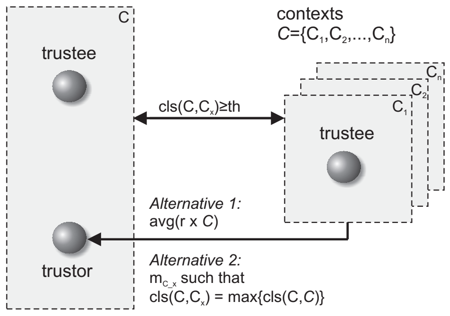
Here, m' can be computed either as $\oplus_k m_k$ (e.g., the consensus among all the trust values) or as the trust value of the entity, amongst $D_1, \ldots, D_N$ that has maximal closeness with B.

4.2 Absent Reputation Information in the Current Context

This section describes the case, where the Trustor wishes to evaluate the trustworthiness of a Trustee so that albeit knowing the Trustee beforehand, the Trustor has no idea of how the Trustee will behave in the current context. The Trustor has the possibility of adopting the same approach as presented above, namely, considering entities which are close enough to the Trustee and utilize their behavior as a guideline for evaluating the



(a)



(b)

Fig. 2. (a) Considering past behavior of similar (e.g., closer than a certain threshold) entities as the Trustee in the current context. The reputation taken is either that of the entity the most similar to the Trustee, or an average reputation among the chosen entities. (b) Considering Trustee's past behavior in similar context(s) as the current one. The reputation is calculated as the reputation of the Trustee in contexts that are close enough (e.g., closer than a certain threshold) to the current context. Either the Trustee's reputation in the closest context, or the average of the reputations among the selected contexts is then chosen.

truswothiness of the Trustee (Figure 2 (a)). However, it is envisaged that often more appropriate results can be obtained by considering the Trustee itself, and its behavior in contexts which are similar enough with the current one (Figure 2 (b)).

Let us continue with the scientific conference example, but this time from the conference chair's point of view. Suppose that the chair is gathering a program committee for the new conference. Here subject areas of the conference call for papers constitute the relevant attributes, which guide the conference chair in inviting appropriate members for the program committee. More specifically, the chair has two major options: In the case of previous conference chair experience in similar enough conferences, the chair can go about evaluating the performance of the PC members in those conferences and make up his mind based on that. Alternatively, the chair can look up other good and similar enough conferences, and count the most frequent PC members and invite them to join.

From a formal point of view, the previous solution is expressed by the following additional rule (as part of REPUTATION-RULES):

$$\frac{A \circ\!\xrightarrow[\mathcal{C}']{dt;(i,m)} B \qquad [\mathcal{C}' \equiv \mathcal{C}]}{A \circ\!\xrightarrow[\mathcal{C}]{dt;(i,m')} B} \tag{10}$$

Again, $\equiv$ can be an equivalence relation, or a reflexive and symmetric relation among contexts that defines the concept of context neighborhood. Figure 2(b) suggests one implementation of relation $\equiv$ based on context similarity; $\mathcal{C} \equiv \mathcal{C}'$ when $\mathtt{cls}(\mathcal{C}, \mathcal{C}')$ is greater than a threshold th. The inferred trust value m', here left unspecified, reasonably depends on m and on the nature of the relationship between $\mathcal{C}'$ and $\mathcal{C}$. For example,

$m' = \omega_s \otimes m$ where $\omega_s = (s, 1-s, 0)$ is the discount build from $s = \texttt{cls}(\mathcal{C}, \mathcal{C}')$ between $\mathcal{C}$ and $\mathcal{C}'$.

Figure 2(b) suggests also a generalization of rule (10); it considers a set of $\equiv$-related contexts where B acted. Formally the rule can be expressed as follows:

$$\frac{\bigcup_{k=1}^{N}\{A \circ\!\xrightarrow[C_k]{dt;(i,m_k)} B\} \qquad [C_k \equiv \mathcal{C}]}{A \circ\!\xrightarrow[\mathcal{C}]{(i,m')} B} \tag{10$'$}$$

Here, m' can be computed either as $\oplus_k m_k$ (e.g., the consensus among all the trust values) or as the trust value of the context that has maximal similarity with $\mathcal{C}$.

5 Indirect Recommendation Information

In rules (6) recommendations carry the context $\mathcal{C}'$ they relate to. Recommendations are considered only if $\equiv$-related with the current context $\mathcal{C}$. Dealing with recommendations in this way is possible only if the Trustor knows the recommenders. We now loosen this requirement. In essence, we allow entities not directly known to the Trustor to be included in the trustworthiness evaluation process as recommenders. In this case, a Trustor may deduce indirect trust directly from an entity, if the entity is "close enough" to the Trustor. In other words, referral trust is approximated by the semantic distance between entities, with the intuitive meaning that "the closer, the more trusted". Formally, this new evaluation step is synthesized by the following variant of rule (6.a):

$$(a)\ \frac{[A \sim D] \quad D \circ\!\xrightarrow[\mathcal{C}]{dt;(i-1,m)} B}{A \circ\!\xrightarrow[\{D\},\mathcal{C}]{it;(i,m')} B} \quad m' \leq m \tag{6.a$'$}$$

Here, the calculus of m' depends on the nature of the relation between A and D; for example, $m' = \omega_s \otimes m$ where ω_s is the discount $(s, 1-s, 0)$ that reflects the closeness s between A and D. The relative importance of a given recommender is estimated based on its relation with the Trustor.

The closeness between two entities can be grounded on the number of links between the Trustor and the recommender. Figure 3 depicts this. Note that there can be multiple parallel paths from the Trustor to the recommender, and they can be taken into account in differing ways. Only the shortest path can be considered, or alternatively all (or some reasonable amount of the) paths can be included in the calculation. The underlying idea is that the more paths there are between the Trustor and the recommender and the shorter they are, the more relevant the recommender is in the eye of the Trustor. Closeness is expressed by the following formula:

$$\texttt{cls}(A, D) = \sum_{k \in I} \frac{1}{\sqrt[\sharp|p_k|]{|p_k| + 1} \cdot k}$$

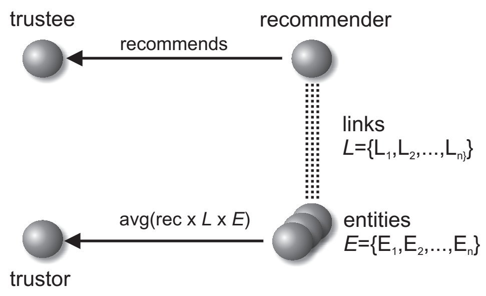


Fig. 3. Considering the opinion of a recommender unknown to the Trustor, but connectable to entities known to the Trustor.

where $p_1, p_2, ..., p_n \in P$ is the ordered set of alternative parallel paths found between the Trustor and the recommender so that $|p_1|$ indicates the number of links in the shortest path, $|p_2|$ in the second-shortest, and so on.

Note that there can be multiple paths that have the same amount of links. As a representative for each set of paths that have an equal amount of links we choose the path with the smallest index. An ordered set of indexes I is created so that only the indexes of the representatives $\in I$. If all paths $\in P$ have a different amount of links, then $I = \{1, \ldots, n\}$. With $\sharp|p_k|$ (for all $k \in I$) we mark the number of paths, in the set of equal length paths, represented by p_k.

As an illustrative example, consider again the conference chair as Trustor A and the proposed PC member's colleague or boss as recommender D and two paths between them. One of the paths has one link and the other two. (If only the shortest path was considered, the closeness metric of the recommender would be $\frac{1}{2 \cdot 1} = .5$). The closeness metric taking into account both paths is $\frac{1}{2 \cdot 1} + \frac{1}{3 \cdot 2} = \frac{2}{3} \approx .67$, and $I = \{1, 2\}$. If we add yet another path to the picture, this time with five links, the closeness metric is $\frac{1}{2 \cdot 1} + \frac{1}{3 \cdot 2} + \frac{1}{6 \cdot 3} = \frac{13}{18} \approx .72$. Here $I = \{1, 2, 3\}$. Now, consider there are three paths between the Trustor A and Trustee D, two having one link each and one having five links. The set of indexes becomes $I = \{1, 3\}$ and the closeness metric becomes $\frac{1}{\sqrt[2]{2} \cdot 1} + \frac{1}{6 \cdot 3} \approx .76$

Two main approaches concerning different link kinds can be distinguished. In the first of these approaches, all link kinds $L_1, L_2, ..., L_n \in L$—be they based on profession, kin, plain acquaintance, and so on—are considered as equally important with regard to the trustworthiness evaluation. The second, in turn, makes distinctions between different link kinds and values some over others. For example, with regard to the program committee membership, professional links can be put more emphasis than acquaintanceships or family relations.

To make distinctions between different link kinds $\in L$ we add a weighting to them. Let $w_{p_{k_j}} \in \mathbb{R}$ be a weighting for a link in path p_k, where $j = 1, \ldots, |p_k|$. The mean link weight for path p_k is defined as

$$W_{p_k} = \frac{\sum_{j=i}^{|p_k|} w_{p_{k_j}}}{|p_k|}.$$

For a set of paths $P = p_1, p_2, \ldots, p_n$, we normalize the path weights W_{p_k} to $[0,1]$ as follows:

$$W'_{p_k} = \frac{W_{p_k}}{max\{W_{p_j} \mid j = 1, \ldots, n\}}.$$

In case there are paths that have an equal amount of links, the mean of their normalized path weights is used. Finally, the weighted closeness metric $\texttt{wcls}(A, D)$ becomes

$$\texttt{wcls}(A, D) = \sum_{k \in I} \frac{W'_{p_k}}{\sqrt[\sharp|p_k|]{|p_k| + 1} \cdot k}$$

Let us continue with the conference example. Suppose we have the same two paths between the Trustor A and recommender D as earlier. But now the shorter path consists of one link of type “family relation”, weighted at 2.5, whereas the path with two links consists of professional links with corresponding weights 4 and 6. The mean link weight for the shorter path is 2.5, and 5 for the longer path. The normalized link weights are thus $\frac{1}{2}$ and 1, respectively. The weighted closeness metric of these paths is $\frac{\frac{1}{2}}{2 \cdot 1} + \frac{1}{3 \cdot 2} = \frac{5}{12} \approx .42$. Suppose that at a later time the family member whose relation was weighted at 2.5 becomes an assistant, and the weight of this relation is 4. In this case the weighted distance metric of these paths becomes $\frac{\frac{4}{5}}{2 \cdot 1} + \frac{1}{3 \cdot 2} = \frac{17}{30} \approx .57$.

6 Conclusions and Future Work

We described and formalized means for evaluating trustworthiness in cases where the Trustor does not possess direct information about the Trustee. We considered both the absence of direct reputation information, that is, lack of Trustor’s personal experiences of the Trustee, and the absence of direct recommendation information, that is, lack of recommendations transmitted to the Trustor by entities known to the Trustor. We discussed cases where the Trustee/Recommender is unknown to the Trustor across contexts, meaning that the Trustor has no knowledge whatsoever about the actions taken by the Trustee/Recommender. In addition, we considered cases where the Trustor has some knowledge about the Trustee/Recommender, but not in the current context.

As a solution we propose to use measures of similarities among entities, and among contexts. Similar entities to the Trustee and a recommender can be used instead, in case Trustee and/or recommenders are unreachable to the Trustor. Additionally, the Trustor can search for a Trustee’s reputation in a similar context, if information concerning the Trustee’s reputation in the present context is missing. Whilst formalizing our approach, we illustrated its usage via a running example.

Our future work around the area includes further investigating the relationships between the Trustor and the Trustee. Research questions are for example comparing different similarity metrics connecting the Trustor with the Trustee (via multiple paths containing recommenders and other acquaintances, as well as varying contexts). In addition, we plan to empirically test and evaluate these metrics.

7 Acknowledgements

This work has been supported by the EU-ITEA project “Trust4All”. In addition to the project consortium and financiers, the authors would like to thank the anonymous reviewers of the 2nd International Semantic Web Policy Workshop (SWPW’06), as well as Sami Kauppinen for his useful comments on the final version of the paper.

References

[1] Shmatikov, V., Talcott, C.: Reputation-based trust management. Journal of Computer Security **13**(1) (2005) 167–190

[2] Dey, A., Salber, D., Abowd, G.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction (HCI) Journal **16**((2-4)) (2001) 97–166

[3] Brézillon, P.: Focusing on context in human-centered computing. IEEE Intelligent Systems **18**(3) (2003) 62–66

[4] Martin, D.: Putting web services in context. Electronic Notes in Theoretical Computer Science **146**(1) (2006) 3–16

[5] Guha, R., McCool, R., Fikes, R.: Contexts for the semantic web. In McIlraith, S., Plexousakis, D., van Harmelen, F., eds.: Proceedings of the International Semantic Web Conference (ISWC 2004). Volume 3298 of Lecture Notes in Computer Science., Berlin, Germany, Springer-Verlag (2004) 32–46

[6] Mostefaoui, S., Hirsbrunner, B.: Context aware service provisioning. In: Proceedings of the IEEE/ACS International Conference on Pervasive Services (ICPS 2004), IEEE (2004) 71–80

[7] Robinson, P., Beigl, M.: Trust context spaces: An infrastructure for pervasive security in context-aware environments. In Hutter, D., et al., eds.: Security in Pervasive Computing, First International Conference, Boppard, Germany, March 12-14, 2003, Revised Papers. Volume 2802 of Lecture Notes in Computer Science., Springer (2004) 157–172

[8] Toivonen, S., Denker, G.: The impact of context on the trustworthiness of communication: An ontological approach. In Golbeck, J., Bonatti, P.A., Nejdl, W., Olmedilla, D., Winslett, M., eds.: Proceedings of the Trust, Security, and Reputation on the Semantic Web workshop, held in conjunction with the 3rd International Semantic Web Conference (ISWC 2004), Hiroshima, Japan, November 7, 2004. Volume 127 of CEUR Workshop Proceedings., CEUR-WS.org (2004)

[9] Toivonen, S., Lenzini, G., Uusitalo, I.: Context-aware trust evaluation functions for dynamic reconfigurable systems. In: Proceedings of the Models of Trust for the Web workshop (MTW’06), held in conjunction with the 15th International World Wide Web Conference (WWW2006) May 22, 2006, Edinburgh, Scotland. CEUR Workshop Proceedings, CEUR-WS.org (2006)

[10] Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: Proc. of the 1996 IEEE Symposium on Security and Privicay, Oakland, CA, USA, 6-8 May 1996, IEEE Computer Society (1996) 164–173

[11] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs, NJ (1995)

[12] Hulsebosch, R.J., Salden, A.H., Bargh, M.S., Ebben, P.W.G., Reitsma, J.: Context sensitive access control. In: Proc. of SACMAT ’05: Proceedings of the 10th ACM symposium on Access control models and technologies (SACMAT 05), June 1-3, 2005, Stockholm, Sweden, ACM Press (2005) 111–119

[13] Bhatti, R., Bertino, E., Ghafoor, A.: A trust-based context-aware access control model for web-services. Distributed and Parallel Databases **18**(1) (2005) 83–105

[14] da Almendra, V., Schwabe, D.: Real-world trust policies. In Kagal, L., Finin, T., Hendler, J.A., eds.: Proc. of the International Semantic Web and Policy Worshop, Nov. 7, 2005, Galway, Irland, UMBC eBiquity (2005)

[15] Yahalom, R., Klein, B., Beth, T.: Trust relationships in secure systems–A distributed authentication perspective. In: Proc of the IEEE Computer Society Symposium on Research in Security and Privacy, May 24-26, 1993, Oakland, California, IEEE Computer Society (1993) 150–163

[16] Beth, T., Borcherding, M., Klein, B.: Valuation of trust in open networks. In: Proc. 3rd European Symposium on Research in Computer Security – ESORICS, Brighton, UK, November 7-9, 1994. Volume 875 of LNCS., Springer (1994) 3–18

[17] Jøsang, A., Gray, L., Kinateder, M.: Simplification and analysis of transitive trust networks. Web Intelligence and Agent Systems Journal **4**(2) (2006) 139–161

[18] Wang, Y., Singh, M.P.: Trust representation and aggregation in a distributed agent system. In: Proc. of the Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI), July 16-20, 2006, Boston, Massachusetts, USA, AAAI Press (2006)

[19] Richardson, M., Agrawal, R., Domingos, P.: Trust management for the semantic web. In Fensel, D., Sycara, K.P., Mylopoulos, J., eds.: Proc. of the International Semantic Web Conference (ISWC 2003), Sanibel Island, FL, USA, 20-23 October 2003. Volume 2870 of LNCS., Springer-Verlag (2003) 351–368

[20] O'Hara, K., Alani, H., Kalfoglou, Y., Shadbolt, N.: Trust strategies for the semantic web. In volbeck, J., Bonatti, P.A., Nejdl, W., Olmedilla, D., Winslett, M., eds.: Proceedings of the Trust, Security, and Reputation on the Semantic Web workshop, held in conjunction with the 3rd International Semantic Web Conference (ISWC 2004), Hiroshima, Japan, November 7, 2004. Volume 127 of CEUR Workshop Proceedings., CEUR-WS.org (2004)

[21] Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems (2005) (available on line on ScienceDirect) in press.

[22] Ruohomaa, S., Kutvonen, L.: Trust management survey. In: Proceedings of the iTrust 3rd International Conference on Trust Management, 23–26, May, 2005, Rocquencourt, France. Volume 3477 of LNCS., Springer-Verlag (2005) 77–92

[23] Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. In Society, I.C., ed.: Proc. of the 334rd Hawaii International Conference on System Sciences (HICSS33), (CD/ROM), Maui, Hawaii, 4-7 January 2000. Volume 6 of HICSS Digital Library., IEEE Computer Society (2000) 1–9

[24] Jøsang, A.: A logic for uncertain probabilities. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **9**(3) (2001) 279–312

[25] Grandison, T., Sloman, M.: A survey of trust in internet applications. IEEE Communications and Survey, Forth Quarter **3**(4) (2000) 2–16

[26] Martinelli, F.: Towards an integrated formal analysis for security and trust. In Steffen, M., Zavattaro, G., eds.: Proc. of the Formal Methods for Open Object-Based Distributed Systems, 7th IFIP WG 6.1 International Conference, FMOODS 2005, Athens, Greece, June 15-17, 2005. Volume 3535 of LNCS., Springer-Verlag (2005) 115–130

[27] Theodorakopoulos, G., Baras, J.S.: Trust evaluation in ad-hoc networks. In: Proc. of the 2004 ACM workshop on Wireless security (WiSe 2004), 1st October 2004, Philadelphia, PA, USA, New York, NY, USA, ACM Press (2004) 1–10

[28] Bistarelli, S.: Semirings for Soft Comstraint Solving and Programming. Volume 2962 of LNCS. Springer-Verlag (2004)

The Virtuous Circle of Expressing Authorization Policies

David Chadwick[1], Angela Sasse[2]

[1] Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK
[2] Computer Science, University College London, Gower St, London, WC1E 6BT, UK

Abstract. This short paper reports on a current project to conduct a detailed investigation into non-security professionals' vocabulary and understanding of e-infrastructure and assets, with the longer term aim of building an ontology and controlled natural language interface that will allow them to build security policies, incorporating complex concepts such as delegation of authority, separation of duties (SoD), obligations and conditions. The interface is designed around the principle of the virtuous circle, whereby the user's controlled natural language input is converted into machine processable XML, and then converted back again into natural language, so that the user can compare the computer's understanding of his policy with his own. The user can then iteratively alter his policy until the input and output are semantically the same. To date, two GUI interfaces have been constructed that aid users in the construction of authorization policies, and produce natural language output. This will serve as a benchmark for measuring the ease of use and effectiveness of the controlled natural language interface. Work has started on the controlled natural language interface, and the first results are reported.

Keywords: Authorization, Policies, Controlled natural language, Virtuous Circle, XML.

1. Introduction

If web services and Grids are to become widely used, they need to be accessible to their target research communities, be secured well enough to be available as needed and function reliably. A key element in realising this ambition is that the owners and donors of web services need to retain control of their resources, and ensure their availability and integrity. To do this, resource owners need to express their policies for who can use their resources, and how. This is termed authorization. Saltzer and Schroeder define authorization as "grant(ing) a principal access to certain information" [1]. Several things are needed to ensure that the authorization policy that the owner intended to be enacted, is the policy that will finally be implemented by the resource's PDP (policy decision point). Firstly resource owners need to be able to state their security requirements correctly and efficiently. In the world of work, this is done through written security policies. In today's computer systems, this is typically done via command line or graphical user interfaces (GUIs) which use specialised security terminology. However, as [3] points out, many computer resources owners fail to even approach this task because they cannot translate their knowledge of resources and access into the computer security terminology used in the GUIs. Secondly, the policy's author needs to be assured that the policy recipients have received the policy and have interpreted it correctly, and behave as intended. In the world of work, policies are circulated to all employees and contractors of a business, in the expectation that they will read and obey them. In the case of computing resources, the policy generated by the interface is translated into a machine processable format, and transferred to the resource PDP for it to enforce. Thirdly, the policy owner needs to periodically check that the policy is indeed being enforced. In the world of work, this might be through periodic reports, audits or spot checks. For computer resources, the PDP will typically write its access control decisions to an audit log that can be periodically inspected by the IT staff. In this way, it is possible to belatedly check, after the fact, that the actual users who eventually gained access to the resource were exactly equal to those that the resource owner intended them to be. A fourth – and key – aspect in the policy specification and enactment process is that the resource owner did not make a mistake in specifying the policy in the first place. By "mistake" we mean that unintended consequences arise from enforcement of the policy (granting access to those who should not have it, or denying access to those who should). These mistakes are caused by misconceptions (in human error terminology) as opposed to errors in executing an intended policy incorrectly, e.g. through typing errors or confusing resource names ("slips" or "lapses" in human error terminology) [4]. When policies are written in controlled natural language, the scope for misconceptions is much reduced, and slips or lapses are more easily detected. Misconceptions can be due to the complexity of the policy, and the likelihood of specifying ambiguities or mutually exclusive clause. The audit log is currently the only (post-facto) way of determining if mistakes were made in the policy specification, as well as in its enforcement. Something better is needed, namely a pre-facto way of determining if the policy specification is correct before enforcement starts.

2. The Virtuous Circle

When specifying web services and grid authorization policies, ideally we want the policy tool to support the resource owner in the entire process of correct policy specification, and improve his/her understanding of access policies as a result of repeated interaction and feedback. In figure 1 below we show a 'virtuous circle' in which the computer system itself helps the user:

- to specify a correct policy,
- ensure that this is the policy that the user intended to specify, and then
- to confirm to the user that this is the policy that will finally be implemented by the PDP.

In figure 1, the user starts with a mental concept of the policy that he intends to enact, and the first step is to transcribe this into the written word in natural language. The language and vocabulary used to describe the policy are underpinned by an ontology that we are currently developing. The natural language policy is then parsed and processed by the computer and converted into a machine understandable policy, written in XML. We have chosen to use XML, since several policy decision points (PDPs) already exist that can read in XML policies and enforce them e.g. XACML [7] and PERMIS [6]. The XML is then processed with an XSL stylesheet, converted back into natural language and displayed to the user. The display not only shows the machine's understanding of the policy, but also is capable of printing out diagnostic error and warning messages to show the user where his policy is wrong, inconsistent or contains superfluous elements. This allows the user to compare the machine's understanding of his policy with his own, and also to correct the errors in his policy.

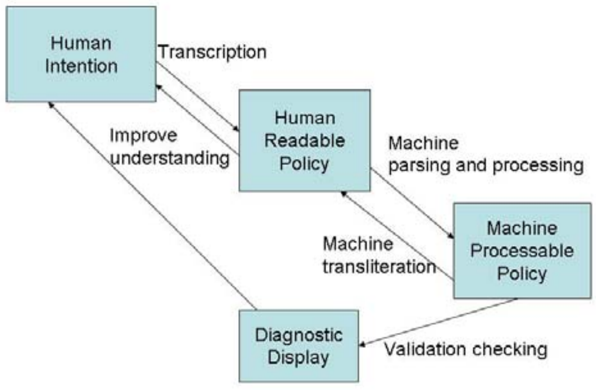


Figure 1. The Virtuous Circle of Policy Specification

3. Progress to Date

To date, we have captured a basic security ontology from interactions with the user community, and built a graphical user interface (GUI), the Policy Editor (see Figure 2) that allows a user to specify a basic authorization policy using this vocabulary. A fuller description of this can be found in [2]. We also have a Policy Wizard variant (see Figure 3) that takes the user step by step through the process of creating a policy, using individual windows from the Policy Editor. The Wizard allows the user to easily create several flavours of a basic authorization policy, but editing an existing policy or adding additional features to a basic policy created by the Wizard, is achieved via the main Policy Editor.

Both GUI tools have screens which display the final policy in either natural language (Figure 4) or XML. The natural language display enables the user to validate, in terms of his own understanding, what the policy actually means to the computer system. This forms the second half of the virtuous circle shown in Figure 1.

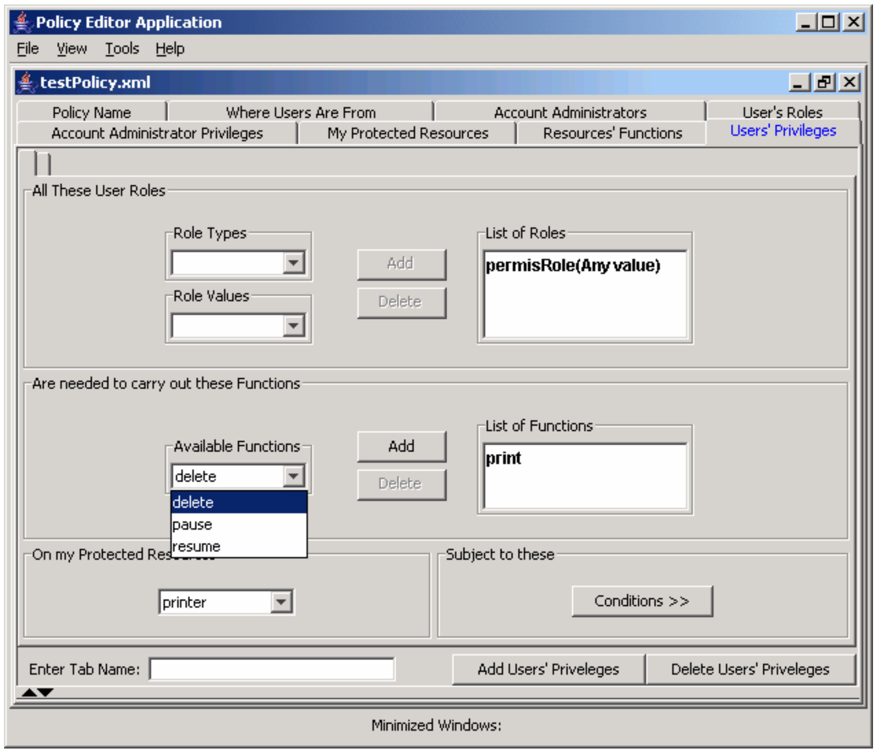


Figure 2. The GUI Policy Tool

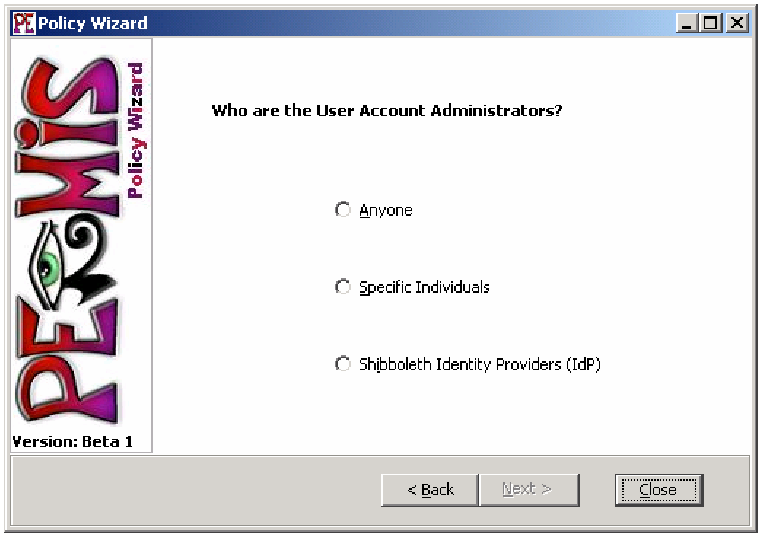
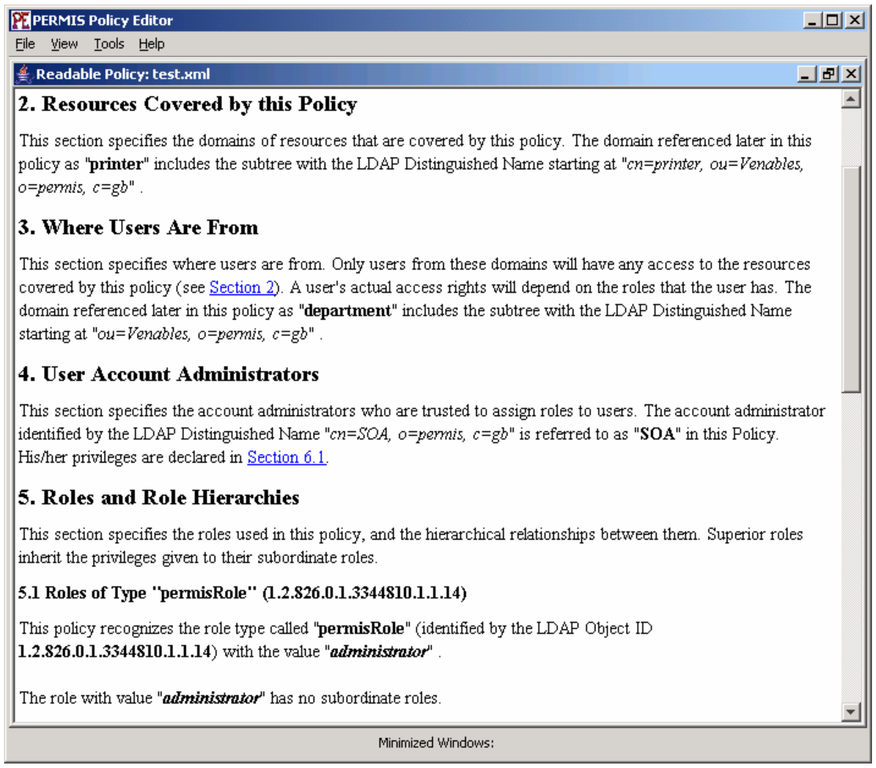


Figure3. The Policy Wizard



2. Resources Covered by this Policy

This section specifies the domains of resources that are covered by this policy. The domain referenced later in this policy as "**printer**" includes the subtree with the LDAP Distinguished Name starting at "*cn=printer, ou=Venables, o=permis, c=gb*" .

3. Where Users Are From

This section specifies where users are from. Only users from these domains will have any access to the resources covered by this policy (see Section 2). A user's actual access rights will depend on the roles that the user has. The domain referenced later in this policy as "**department**" includes the subtree with the LDAP Distinguished Name starting at "*ou=Venables, o=permis, c=gb*" .

4. User Account Administrators

This section specifies the account administrators who are trusted to assign roles to users. The account administrator identified by the LDAP Distinguished Name "*cn=SOA, o=permis, c=gb*" is referred to as "**SOA**" in this Policy. His/her privileges are declared in Section 6.1.

5. Roles and Role Hierarchies

This section specifies the roles used in this policy, and the hierarchical relationships between them. Superior roles inherit the privileges given to their subordinate roles.

5.1 Roles of Type "permisRole" (1.2.826.0.1.3344810.1.1.14)

This policy recognizes the role type called "**permisRole**" (identified by the LDAP Object ID **1.2.826.0.1.3344810.1.1.14**) with the value "***administrator***" .

The role with value "***administrator***" has no subordinate roles.

Figure 4. The Policy Specified in Natural Language

Note that the policy is displayed to the user in full natural language using an XSL style sheet. In the next stage of the project the user will be allowed to create a new authorisation policy using controlled natural language.

4. Controlled Natural Language

Natural language processing (NLP) is very hard due to the ambiguities and complex structure of natural language. Machine translation has been continuously refined for decades. Major industry leaders are still performing research into machine translation, paraphrasing and information extraction [8, 9]. However, they provide no free tools to academia. A number of universities in the UK are developing NLP and Information Extraction technologies. However, they are mostly directed at annotating scientific texts and analysing vast sources of information in natural language to spot pieces of text that are of interest to a scientist [12, 13].

In controlled natural language either the vocabulary and/or grammar that can be used are limited, being a subset of natural language. This makes machine processing much easier and more tractable than using free form natural language. The GATE project (http://gate.ac.uk/), lead by the University of Sheffield, has produced a natural language processing kit [5]. It includes a set of tools and grammars that allow English and other texts to be analysed. In the project SEKT (Semantic Knowledge Technology – http://www.sekt-project.com/), the GATE team has investigated the application of controlled languages [10] to the provision of natural language interfaces for tasks such as web service protocol description or ontology construction. Sheffield has developed a Controlled Language Information Extraction [11] tool (CLIE) to aid users in their task. The number of sentence structures allowed by CLIE is very limited, which means that it is very easy to learn to use, much easier than for example OWL or RDF or tools such as Prodigy. However, the vocabulary for classes and instances is unlimited, which means that complex ontologies can still be created.

We are experimenting with using CLIE to construct ontologies for authorization policies. CLIE supports three sentence constructs for creating class and instance hierarchies. "There are <class>", "<subclass> is a type of <superclass>" and "<Instance> is a <class>". CLIE supports one sentence construct for defining relationships between classes "<class> (can) have <class>", and a similar one for adding properties to classes "<class> (can) have textual <property name>". CLIE supports two sentence constructs for setting property values for instances: "<instance> has <instance>" and "<instance> has property <property> with value <instance>". This limited language allows us to reproduce nearly all the functionality

There are policies.
"My AC policy" is a policy.
There are resources and users.
David is a user.
Printer is a type of resource.
"HP Laserjet4" is a printer.
There are domains.
Kent is a domain.
There are "User Account Administrators".
Peter is a User Account Administrator.
There are actions and parameters.
Print is an action.
Delete is an action.
Pause and resume are actions.
"No of pages" is a parameter.
Actions have parameters.
Print has action with value "No of pages".
There are roles.
Student is a role.
Staff is a role.
Resources have actions.
"HP Laserjet4" has action with value print.
"HP Laserjet4"has action with value delete.
"HP Laserjet4" has action with value pause.
"HP Laserjet4" has action with value resume.

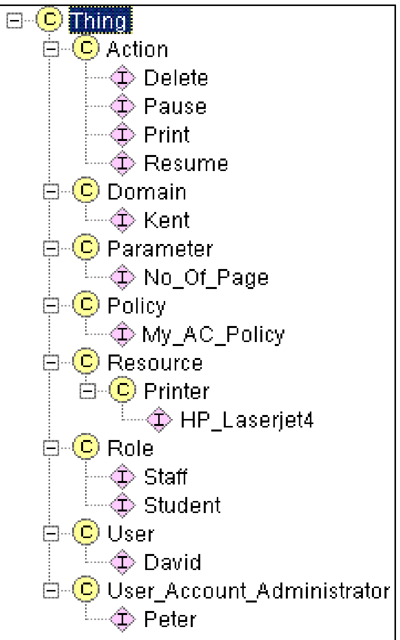


Table 1. An Example Authorisation Policy Ontology using CLIE

of 6 of the 8 tabs in our Policy Editor. (The two tabs that currently cannot be specified are the Account Administrator Privileges and the Users' Privileges.) The left hand column of Table 1 shows the policy sentences typed in by the user, and the right hand column shows the resulting class-instance hierarchy ontology created by CLIE (note that properties and instance property value assignments are not shown in the right hand column).

5. Future Work

CLIE is a good start but still needs some enhancements. A current limitation is that it only allows the "(can) have" relationship between classes and instances. We have a requirement to specify new types of relationship, such as the superior/subordinate relationships in role hierarchies, and the "can assign" relationship between administrators and roles. Sometimes simpler sentence constructs would be more user friendly, for example, making "with value" optional when setting property values. More complex sentence constructs are also needed such as "users with these properties can access resources with these properties providing these conditions are met". This will require us to tailor the GATE software to use the specific grammar of these authorisation sentences using the base ontology provided by CLIE. GATE creates an in-memory semantic representation of the user's newly created sentences using the ontology. Any unrecognised or erroneous words will be highlighted, prompting the user to take some clarifying action. Plugin tools will be developed to extract the user's intended semantics from unrecognised words, from partially specified conditions (e.g. If later than 5), ambiguous phrases (e.g. double negatives) or conflicting semantics (e.g. employees can print but managers cannot, when managers are superior to employees). Finally, the in-memory semantic representation of the policy will be compiled into two XML authorisation policy languages (XACML and PERMIS) so that they can then be displayed in natural language via style sheets, and subsequently read into their respective PDPs for access control decision making.

Turning to the GUI tools, we are currently increasing the ontology used in them and incorporating more complex security concepts such as separation of duties, mutually exclusive roles, obligations and constraints. This is being done by analysing transcript's from interviews with researchers recorded during previous e-Science research projects, collected over the past few years. The extracted ontology will be validated by testing it with potential e-Science users. The users will be asked to specify polices in natural language for a number of scenarios set in their own research environments, and to interpret a number of semi-structured policies and queries.

The final evaluation trial is planned to be conducted with e-Science researchers from a variety of projects – medicine and bioinformatics, sciences, social science, data grids and computational grids. They will be asked to carry out a set of standardised policy specification tasks using the natural language tool and the existing GUI tools. With participants' permission, the interactions will be recorded (both on the system and with a video camera), and participants will be asked to *think aloud* while carrying out the tasks and when interpreting feedback received from the tools.

This will be followed by a brief questionnaire to assess user satisfaction and perceived user effort. The ease of use and effectiveness of the natural language interface will be compared and contrasted with that of the GUI and Wizard interfaces.

References

1 Saltzer, J.H., and Schroeder, M.D. "The Protection of Information in Computer Systems," Proceedings of IEEE, 63(9), 1278-1308, 1975.
2 Brostoff, S., Sasse, A., Chadwick, D., Cunningham, J., Mbanaso, U., Otenko, S. "*"R-What?"* Development of a Role-Based Access Control (RBAC) Policy-Writing Tool for e-Scientists" to appear in Software Practice and Experience, 2005
3 Barman, S. "Writing Information Security Policies". New Riders 2001
4 Reason, J. "Human Error". Wiley 1990.
5 H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02).* Philadelphia, July 2002.
6 D.W.Chadwick, A. Otenko, E.Ball. "Role-based access control with X.509 attribute certificates", IEEE Internet Computing, March-April 2003, pp. 62-69.
7 OASIS "eXtensible Access Control Markup Language (XACML) Version 2.0" OASIS Standard, 1 Feb 2005
8 Quirk, C., Brockett, C., and Dolan, W.B. "Monolingual Machine Translation for Paraphrase Generation", In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 25-26 July 2004, Barcelona Spain, pp. 142-149.
9 Jennifer Chu-Carroll, Krzysztof Czuba, John Prager, and Abraham Ittycheriah, "In Question Answering, Two Heads are Better Than One", Human Language Technology Conference (HLT/NAACL), 2003
10 S. Pulman. Controlled Language for Knowledge Representation. In *CLAW96: Proceedings of the First International Workshop on Controlled Language Applications*, pages 233–242, Leuven, Belgium, 1996.
11 Valentin Tablan, Tamara Polajnar, Hamish Cunningham, Kalina Bontcheva. "User-friendly ontology authoring using a controlled language". Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC), Genoa, Italy, May 2006
12 Briscoe, E. J. and J. Carroll. "Robust accurate statistical annotation of general text". In *Proceedings of the 3rd International Conference on Language Resources and Evaluation* (LREC 2002).
13 Alex Morgan, Lynette Hirschman, Alexander Yeh, and Marc Colosimo. "Gene name extraction using FlyBase resources". In Sophia Ananiadou and Junichi Tsujii (eds.), *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine.*

Aligning WSMO and WS-Policy [⋆]

Dumitru Roman, Jacek Kopecký, Ioan Toma, and Dieter Fensel

Digital Enterprise Research Institute, Innsbruck, Austria
{`firstname.lastname`}`@deri.org`

Abstract. Service-Oriented Architectures (SOAs) suggest that IT systems should be developed from coarse-grained, loosely coupled, business-aligned components, so called services. One way towards loose coupling is to refrain from hard-coding policies in the system and to represent them explicitly. Semantic Web Services (SWS) add semantics to Web services (main realization of SOAs). However, SWS research currently ignores most of the work done on Web service policies, therefore in this paper we present a proposal for combining WSMO, a major SWS framework, and WS-Policy Framework, a set of specifications with heavy industrial backing. The resulting combination is aimed at serving as the basis of applying the logical reasoning capabilities that have been developed (or are being developed) for WSMO to WS-Policy, and the basis for integrating WSMO in the WS-Policy framework.

1 Introduction

Semantic Web based approaches to policies have been recently advocated (in e.g. [2, 16, 11, 7, 8, 15]) in order to cope with some of the limitations of traditional policy handling systems (e.g. policy systems existing before the era of Web services) where the policies are hard coded into a system according to the functional requirements, language features, and design decisions, without separating policy specification from policy implementation, and thus making difficult and expensive to change and enforce policies. Policy *specification*, *enforcement* and *revision* are three basic tasks a policy-handling system must provide. In this paper we focus on issues related to policy specification in semantically enriched service-oriented environments. Policy specification languages enable policies to be captured independent from a concrete system implementation. Such languages are to be interpreted by a policy engine at runtime, which makes dynamic policy changes possible; they formalize the intent of the designer into a form that can be analyzed and interpreted by a policy-aware system.

In this context, we are interested in identifying the potential relations between the policy languages and the so-called Semantic Web Services [10, 6]. Semantic Web Services have emerged as a combination of Semantic Web and Web services technologies with the aim of automating different service-related tasks, such as Web service discovery, composition, mediation, or execution. Several proposals have been put forward in this area.[1] In this paper we choose the WSMO approach [13] for analysis of its relation to policy domain because, in contrast to other approaches, it provides a unifying

⋆ The work is funded by the European Commission under the projects ASG, DIP and SWING.
[1] We refer the reader to [12] for a detailed discussion on the SWS approaches.

framework combining a conceptual model (Web Service Modeling Ontology), a formal syntax and language (Web Service Modeling Language), and an execution environment (Web Service Execution Environment).

Common denominators have been identified between policies that may be leveraged to improve policy uniformity and streamline service-oriented enterprise-wide policy implementation. In particular, a set of new enforcement policies were proposed in the industry using novel policy concepts together with building blocks from e.g. XACML, WS-Policy and SAML. For the purpose of this paper, we have chosen to use the Web Services Policy Framework (WS-Policy) [5], a general purpose model and a syntax to describe the policies of a Web Service. Our choice is motivated by the fact that WS-Policy has significant industry backing. Interested parties are already creating policy assertions for various domains and the major commercial Web services infrastructure stacks already have or are building support for policy-based Web service invocations.

The aim of this paper is to investigate the potential relations between the WSMO approach to SWS and the WS-Policy framework, and to offer a basis for further research on the use of policies in the context of semantically enriched service environments. Issues like policy conformance checking, policy-based semi-automatic negotiations between users and services, or policy-based matchmaking etc., are important in our context, however they are out of scope of this paper. The scope of this paper is limited to identifying the commonalities and differences between the conceptual models of WSMO and WS-Policy, and proposing a concrete way for combining them.[2]

Despite the differences in terminology, we can say that both WSMO and WS-Policy describe the capabilities and constraints of Web services. On one side, WSMO uses very clearly defined terms like *Web service*, *Capability*, *Interface* etc. to capture all the relevant properties of Web services. On the other side, WS-Policy talks about generic assertions, focusing on their combinations in whole policies, and then attaching these policies to various subjects, among which Web service endpoints are especially relevant for our work.

We can identify two generic concepts from the WS-Policy framework that could encompass elements from WSMO: Policy Assertion and Policy Subject. In other words, parts of a WSMO description can be mentioned as assertions in some policy, or a policy could be attached to parts of a WSMO description. We describe these two directions and the syntax realizing the actual connections in the following two sections (Section 2 and Section 3). In Section 4 we briefly highlight related works, and in Section 5 we conclude this paper and point out potential directions for future work.

2 Attaching Policies to WSMO

To analyze how policies can be attached to WSMO, we need first to introduce a distinction between functional and non-functional properties.[3] In any application, the *func-*

[2] Because of the limited space we do not provide an overview of the concepts that WSMO and WS-Policy introduce, however, we refer the reader to [13] and [5] for a detailed introduction to WSMO and WS-Policy, respectively.

[3] A more detailed discussion on functional vs. non-functional properties can be found in the position paper for the W3C Workshop on Constraints and Capabilities for Web Services [1].

tional part of any data contains crucial information necessary for the application to do its job. *Non-functional* properties (NFPs), on the other hand, contain such additional data that may help the application do a better job, or to refine its functionality.

For example, one of the aims of WSMO is Web service discovery (we can see WSMO as an application for service discovery), and to enable discovery, WSMO describes client *Goals* and the *Capabilities* of the available service. *Goals* and *Capabilities* are the the necessary inputs to a matching algorithm, therefore they are functional aspects of WSMO descriptions. While pure Web service discovery only requires the *Capabilities* and *Goals*, the match maker can also take into account preferences and constraints over parameters like the price, availability or quality of a service. Because such parameters are not critical for the match maker, they are modeled as non-functional properties.

The distinction between functional and non-functional parameters depends highly on the application that uses the particular parameter — a functional parameter of one application can be non-functional in another. For instance, Semantic Web Services are an application that automates the use of Web services, and the price of a service is generally modeled as a non-functional property; however a shopping agent application will have price as one of its main functional parameters.

In WSMO, the distinction between functional and non-functional properties is made very clear: WSMO enumerates all the relevant functional properties (for example *Web service* has *Capability* and *Interface* as its functional properties) and it allows an extensible bag of NFPs everywhere. WS-Policy does not have any such distinction, so it can be used to express both functional and non-functional policy assertions, depending on the application that employs policies. Since WSMO enumerates in its conceptual model all the parameters that are functional for the aim of WSMO, policies can be treated as non-functional data, therefore when a WS-Policy is attached to a WSMO element, it is abstractly added to the non-functional properties of that element.

Fig. 1. Attaching Policies to WSMO Elements

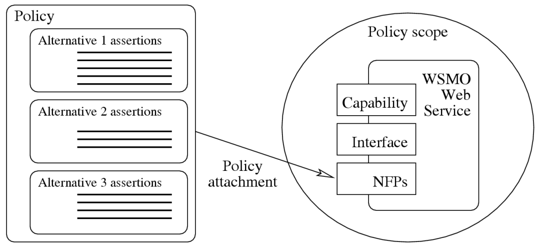


Figure 1 shows how WSMO elements can be used as policy assertions in a policy that is then attached to a particular policy scope. In particular, a policy is attached here to a WSMO Web service description, and it is treated as a non-functional property of that service. Due to the way policy attachment works syntactically, the policy is in fact embedded or referenced from within the non-functional properties block of the Web service description.

To summarize, WSMO elements can serve as WS-Policy Policy Subjects, and any policies attached to those WSMO elements are treated as non-functional properties. In the following, we present the syntax for including policies as non-functional properties in WSMO descriptions. Using this mechanism, the existing and future policy assertions can usefully complement the Dublin Core NFPs [18] currently used by WSMO.

To attach policies generically to XML elements such as WSDL descriptions, the WS-PolicyAttachment specification [4] defines an XML attribute called `PolicyURIs` and an XML element called `PolicyReference`. Both the attribute and the element point to external policies using URIs. WS-PolicyAttachment introduces both of them because some XML languages restrict attribute or element-based extensibility.

In WSMO, we use the namespace-qualified name `wsp:PolicyReference` for an NFP; its value is one or more URIs of policies attached to the owner WSMO element:

```
service ACMEService
  nonFunctionalProperties
    wsp#PolicyReference hasValue
      {_"http://fabrikam123.example.com/policies/DSIG",
       _"http://fabrikam123.example.com/policies/SECTOK"}
  endNonFunctionalProperties
```

In some cases it can be useful for manageability reasons to include the whole policy in the WSMO description, especially when the policy is fairly small. For this purpose we reuse the namespace-qualified name `wsp:Policy` as the name of a non-functional property whose content is the XML serialization of the whole `Policy` element. This is illustrated by the following example:

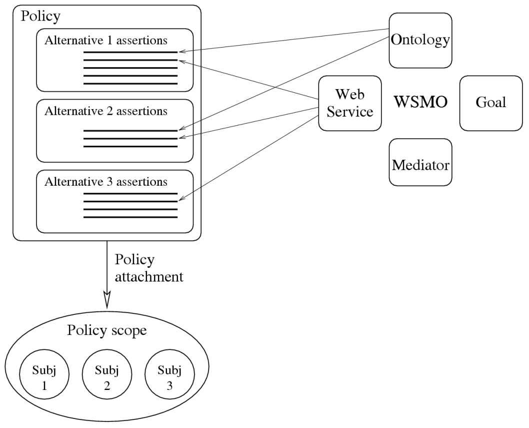
```
service ACMEService
   nonFunctionalProperties
     wsp#Policy hasValue
       "<wsp:Policy>
          <wsp:ExactlyOne>
            <wsse:SecurityToken>
              <wsse:TokenType>wsse:Kerberosv5TGT
              </wsse:TokenType>
            </wsse:SecurityToken>
            <wsse:SecurityToken>
              <wsse:TokenType>wsse:X509v3
              </wsse:TokenType>
            </wsse:SecurityToken>
          </wsp:ExactlyOne>
        </wsp:Policy>"
   endNonFunctionalProperties
```

To summarize, we allow attaching both external and embedded policies to WSMO elements, treating the attached policies as non-functional properties. For this, we reuse the WS-Policy element names (`wsp:PolicyReference` and `wsp:Policy`) as NFP identifiers in WSMO.

3 WSMO as Policy Assertions

WS-Policy is a mechanism of combining domain-specific policy assertions and attaching them to various policy subjects. WSMO descriptions can be viewed as policy assertions and combined with others in policy alternatives. Figure 2 shows how WSMO elements can be used as policy assertions in a policy that is then attached to a particular policy scope, for example a Web service endpoint. Such a policy would thus attach the WSMO Web service description to that endpoint.

Fig. 2. Using WSMO Elements as Policy Assertions



We can envision, for example, a policy that ties the capabilities of a Web service with various security settings. A service may offer some basic functionality with strong authentication but weak communication channel encryption, and more advanced functionality can be available provided that strong encryption is employed.

WSMO currently does not have any specific mechanism for expressing that alternative WSMO descriptions are in effect in conjunction with various non-functional properties,[4] and WS-Policy seems to be a widely-adopted mechanism for expressing exactly such alternatives, therefore even though WSMO descriptions would not normally be treated as policy assertions, such an approach may prove beneficial.

Because policy assertions must be XML elements, we can reuse WSML/XML serialization format (see [3]) for representing WSMO descriptions as policy assertions in WS-Policy. To attach a whole WSMO description as a single policy assertion, we use the element `wsml:wsml`.[5] For finer granularity (e.g. only asserting a single capability description) we can reuse the appropriate elements like `wsml:capability`.

In some situations it may be beneficial only to refer to a WSMO description (as opposed to including it inline as a policy assertion). For referring to a whole WSML file we introduce the element `wsml:descriptionReference` that refers to a WSML document, and similarly we can introduce specific elements for referring to specific elements of WSMO, for instance to refer to a capability or an interface we can use elements `wsml:capabilityReference` and `wsml:interfaceReference` that would refer to the identifiers of the WSMO capability or interface descriptions.

The listing in Figure 3 is a policy that claims the policy subject is described by the WSMO services http://example.org/services/ticketService and http://example.org/services/billingService. Lines 2–7 show a WSML/XML element `wsml:webService` that, in this context, means a policy assertion assigning a WSMO service description defined inline. Similarly, lines 13–15 show a reference to such a description, using the

[4] See Section 2 for discussion of why WSMO treats WS-Policy as non-functional properties.

[5] The first `wsml` is a namespace prefix and the second is the name of the XML element container for WSML/XML syntax.

Fig. 3. Example policy with WSMO webService as policy assertion

```
01  <wsp:Policy>
02    <wsml:webService
03         name="http://example.org/services/ticketService">
04      <wsml:capability name="ticketing">
05        ...
06      </wsml:capability>
07    </wsml:webService>
08    <wsml:ontology
09         name="http://example.org/ontologies/ticketing/">
10      <wsml:concept name="Ticket"/>
11      ...
12    </wsml:ontology>
13    <wsml:serviceReference>
14      http://example.org/services/billingService
15    </wsml:serviceReference>
16  </wsp:Policy>
```

new element `wsml:serviceReference`. Finally, lines 8–12 contain an ontology which is used by the ticketService definition.

The conclusion is that to represent a policy assertion “the policy subject has the following WSMO description” we can use any global WSML/XML element as appropriate, and to represent an assertion only referencing a WSMO description we have to create specific elements for each type of WSMO entity that we want to reference.

4 Related Works

To the best of our knowledge, the approach presented in this paper is the first attempt to combine the WSMO approach to SWS with policies. However, it is worth mentioning other works that deal with combining policies and Semantic Web technologies in general. Such approaches differ from our work in the sense that they are focused on Semantic Web technologies other than WSMO, and some of them are aimed at solving a specific policy-related task (e.g. policy conformance check, matchmaking of Web services, etc.), whereas we are mainly focused in this paper on combining WSMO and WS-Policy, without emphasizing at this stage any particular policy-related task.

We can classify relevant related works as those that deal directly with WS-Policy, and those which take a more general approach to policies, and thus not committing to WS-Policy as a framework for representing policies.

Among works that deal directly with WS-Policy, [9] provides a mapping of WS-Policy to OWL-DL in order to use OWL-DL reasoners to check policy conformance, [17] uses OWL ontologies for creating policy assertions with semantics in WS-Policy in order to enable matching the non-functional properties of Web Services represented using WS-policy, and [14] proposes an approach to behaviour-based discovery of Web Services by which business rules that govern service behaviour are described as a policy, in the form of ontological information; here WS-Policy is used to associate such a policy to the Web Service.

Works that do not deal directly with WS-Policy (but which take into account Semantic Web based approaches to policies) focus more on administrative policies such as security and resource control policies. In this category it is worth mentioning KAoS [16] as one of the first efforts to represent policies using OWL, [11] that discusses how to represent policy inheritance and composition based on credential ontologies and formalizes these representations in Frame-Logic, [15] that proposes a hybrid approach for policy specifications which exploits the expressive capabilities of Description Logic languages and Logic Programming approaches; and Rein [8],[6] which is a framework for representing and reasoning over policies in the Semantic Web.

5 Conclusions and Outlook

This paper is a first step to combine WS-Policy (a policy framework with significant industry backing) and the WSMO approach to Semantic Web Services (one of the most important proposals for Semantic Web Services to date). We identified potential ways of combining them, and provided the necessary syntax. The new syntax elements are summarized in Table 1.

Table 1. Syntax for combining WSMO and WS-Policy

Name	**Type**	**Description**
wsp#PolicyReference	NFP identifier	non-functional property referring to an external policy file by URI
wsp#Policy	NFP identifier	non-functional property containing an XML serialization of a policy
wsml:wsml	XML Element	the root element of WSML/XML syntax, reused as a policy assertion "the embedded WSML description applies (to the policy subject)"
wsml:webService wsml:goal ...	XML Element	other WSML/XML elements reused as a policy assertion "the embedded WSML description applies (to the policy subject)"
wsml:descriptionReference wsml:webServiceReference wsml:goalReference ...	XML Element	policy assertions that refer to WSMO definitions by their identifier URIs

The proposals presented in the paper represent the basis for enabling the use of policies from SWS environments for tasks such as policy-based semi-automatic negotiations between users and services, policy-based matchmaking between users requests and services, scheduling of service compositions under certain constraints, etc; but also for enabling the use of semantic descriptions as policy assertions.

In future work, we plan to investigate concrete applications of the combination of WSMO and WS-Policy presented in this paper, especially applying existing reasoning techniques developed in the context of WSMO to policy specification, as well as developing new techniques in order to provide automated support for Web service discovery, negotiation, selection, composition, invocation and monitoring with policy awareness.

[6] `http://dig.csail.mit.edu/2006/06/rein/`

References

1. S. Arroyo, C. Bussler, J. Kopecký, R. Lara, A. Polleres, and M. Zaremba. Web Service Capabilities and Constraints in WSMO. In *W3C Workshop on Constraints and Capabilities for Web Services*, September 2004.
2. P. A. Bonatti and D. Olmedilla. Semantic Web Policies: Where are we and What is still Missing? A tutorial at ESWC'06, 2006. Available at `http://www.l3s.de/~olmedilla/events/2006/ESWC06/ESWC06_Tutorial.html`.
3. J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language WSML. Available via `http://www.wsmo.org`, October 2005.
4. C. Sharp (editor) et al. Web Services Policy 1.2 – Attachment (WS-PolicyAttachment). Technical note, April 2006. Available via `http://www.w3.org/`.
5. J. Schlimmer (editor) et al. Web Services Policy 1.2 – Framework (WS-Policy). Technical note, April 2006. Available via `http://www.w3.org/`.
6. D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2), 2002.
7. L. Kagal, T. Berners-Lee, D. Connolly, and D. Weitzner. Self-Describing Delegation Networks for the Web. In *Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, pages 205–214, 2006.
8. L. Kagal, T. Berners-Lee, D. Connolly, and D. J. Weitzner. Using Semantic Web Technologies for Policy Management on the Web. In *AAAI*, 2006.
9. V. Kolovski, B. Parsia, Y. Katz, and J. A. Hendler. Representing Web Service Policies in OWL-DL. In *International Semantic Web Conference*, pages 461–475, 2005.
10. S. McIlraith, T. Son, and H. Zeng. Semantic Web Services. In *IEEE Intelligent Systems (Special Issue on the Semantic Web)*, 2001.
11. W. Nejdl, D. Olmedilla, M. Winslett, and C. C. Zhang. Ontology-Based Policy Specification and Management. In *2nd European Semantic Web Conference (ESWC)*, pages 290–302, 2005.
12. D. Roman, J. de Bruijn, A. Mocan, I. Toma, H. Lausen, J. Kopecky, D. Fensel, J. Domingue, S. Galizia, and L. Cabral. Semantic Web Services – Approaches and Perspectives. In J. Davies, P. Warren, and R. Studer, editors, *Semantic Web Technologies: Trends and Research in Ontology-based Systems*, pages 191–236. John Wiley & Sons, 2006.
13. D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1(1):77–106, 2005.
14. N. Sriharee, T. Senivongse, K. Verma, and A. P. Sheth. On Using WS-Policy, Ontology, and Rule Reasoning to Discover Web Services. In *INTELLCOMM*, pages 246–255, 2004.
15. A. Toninelli, J. Bradshaw, L. Kagal, and R. Montanari. Rule-based and Ontology-based Policies: Toward a Hybrid Approach to Control Agents in Pervasive Environments. In *Proceedings of the Semantic Web and Policy Workshop*, November 2005.
16. A. Uszok, J. M. Bradshaw, R. Jeffers, A. Tate, and J. Dalton. Applying KAoS Services to Ensure Policy Compliance for Semantic Web Services Workflow Composition and Enactment. In *International Semantic Web Conference*, 2004.
17. K. Verma, R. Akkiraju, and R. Goodwin. Semantic Matching of Web Service Policies. In *SDWP Workshop*, 2005.
18. S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin Core Metadata for Resource Discovery. RFC 2413, IETF, 1998.

WS-Policy and Beyond: Application of OWL Defaults to Web Service Policies

Vladimir Kolovski[1] and Bijan Parsia[2]

[1] Department of Computer Science,
University of Maryland, College Park, MD USA,
kolovski@cs.umd.edu
[2] School of Computer Science,
The University of Manchester, UK,
bparsia@cs.man.ac.uk

Abstract. Recently, there has been an increased amount of attention dedicated to WS-Policy - it has become a W3C submission and a working group was formed to standardize the specification. In our previous work, we provided a mapping of WS-Policy to OWL-DL. In this paper, we continue that work by analyzing the operation of policy intersection (determining whether two web service policies are compatible). We show how this operation motivates the use of a non-monotonic extension of OWL in the form of OWL default rules. We discuss our prototype implementation of an OWL defaults reasoner based on Baader and Hollunder's terminological defaults.

1 Introduction

Recently, there have been many different web service policy language proposals with varying degrees of expressivity and complexity [21, 6, 1]. One of these languages, WS-Policy became a W3C member submission and is the basis for the WS-Policy working group [3].

In previous work [13] we described a translation of WS-Policy to a standardized logic (OWL-DL). This mapping essentially provided a formal semantics for the framework, and allowed us to use an OWL DL reasoner for policy processing tasks such as determining policy equivalence, incompatibility, containment, incoherence and explanation. In this paper, we provide additional results on the translation by exploring the operation of policy *intersection*. This operation determines whether two policies are compatible and generally involves domain-specific processing. In the official specification of WS-Policy [21], only an approximation algorithm is defined for this operation. Instead, we describe an algorithm based on OWL-DL extended with default rules. Because default logic is computationally more expensive than the logic behind OWL-DL, we do provide clear motivations for our usage of defaults.

To provide reasoning support for OWL defaults we have extended an open-source OWL-DL reasoner (Pellet). Our implementation is based on Baader and Hollunder's

[3] WS-Policy Working Group web site: http://www.w3.org/2002/ws/policy/

terminological default logic [3] (adapted to OWL-DL). To retain decidability, the terminological default logic of Baader and Hollunder restricts the default rules to named individuals only, similar to DL-safe rules. We provide a brief description of our system in Section 6.

2 Preliminaries

In this section we provide brief overview of the WS-Policy framework and Reiter's default logic, which served as the basis of our implementation.

2.1 WS-Policy Framework Overview

The WS-Policy Framework provides a general purpose model and syntax to describe the policies of a Web service. Its scope is limited to allowing endpoints to specify requirements and capabilities needed for establishing a connection. Its initial goal is not to be used as a language for expressing more complex, application-specific policies that take effect after the connection is established. For this purpose, WS-Policy introduces a simple and extensible grammar that consists of *assertions* and *alternatives*.

An assertion is the basic unit of a policy. For example, an assertion could declare that the message should be encrypted. The actual definitions and meaning of the assertions are domain-dependent and not defined in the WS-Policy Framework. An assertion is defined by a unique Qualified Name, and can be a simple string or a complex object with many sub elements and attributes. Note that an assertion can contain a nested policy expression.

A set of assertions is called a policy alternative, and a set of alternatives comprises a policy. For an alternative to be supported by a web service requester, all assertions in that alternative have to be satisfied by that requester. For a policy to be supported by a requester, one or more alternatives need to be supported. Following is a schema outline for the normal form of a policy expression:

```
<wsp:Policy>
   <wsp:ExactlyOne>
      [ <wsp:All> [<Assertion>  </Assertion>]* </wsp:All> ]*
   </wsp:ExactlyOne>
</wsp:Policy>
```

2.2 Default Logic

Reiter's default logic is a nonmonotonic formalism for expressing commonsense rules of reasoning. These rules, called default rules (or simply *defaults*), are of the form:

$$\frac{\alpha \, : \, \beta}{\gamma}$$

where α, β, γ are first-order formulae. We say α is the *prerequisite* of the rule, β is the *justification* and γ the *consequent*. Intuitively, a default rule can be read as: if I can prove the prerequisite from what I believe, and the justification is consistent with what I believe, then add the consequent to my set of beliefs.

Definition 1 *A default theory is a pair $\langle \mathcal{W}, \mathcal{D} \rangle$ where $\mathcal{W}$ is a set of closed first-order formulae (containing the initial world description) and $\mathcal{D}$ is a set of default rules. A default theory is closed if there are no free variables in its default rules.*

Possible sets of conclusions from a default theory are defined in terms of *extensions* of the theory. Extensions are deductively closed sets of formulae that also include the original set of facts from the world description. Extensions are also closed under the application of defaults in $\mathcal{D}$ - we keep applying default rules as long as possible to generate an extension.

Default rules can conflict. A simple example is when two defaults d_1 and d_2 are applicable yet the consequent of d_1 is inconsistent with the consequent of d_2. We then typically end up with two extensions: one where the consequent of d_1 holds, and one where the consequent of d_2 holds.

3 Updated OWL-DL Mapping

In [13] we presented a mapping of WS-Policy to OWL-DL based on the idea that service policy assertions and alternatives were mapped to classes, and web service requesters are mapped to OWL individuals. With this mapping, checking whether a web service requester satisfies a particular policy can then be reduced to simply checking whether the OWL individual representing the requester is a member of the OWL class representing the policy. The mapping was relatively simple since there are only two relevant constructs in a WS-Policy in a normal form (<wsp:exactlyOne>, <wsp:All>). Due to the name of one of the operators (<wsp:exactlyOne>) and the ambiguity in the WS-Policy specifications, we translated it to a logical XOR. Thus the policy $P = ExactlyOne(A,\ B)$ was mapped to the description logic expression: $\mathsf{P} = (\mathsf{A} \sqcup \mathsf{B}) \ \sqcap \neg(\mathsf{A} \sqcap \mathsf{B})$ (<wsp:All> was mapped to logical conjunction).

However, due to the open world assumption present in OWL-DL, our previous mapping produces non-intuitive results. For example, if a request r comes in such that $r : A$, and the policy P contains only two alternatives, A and B, we will not be able to infer that the request r satisfies P (i.e., r is of type $(\mathsf{A} \sqcup \mathsf{B}) \ \sqcap \ \neg(\mathsf{A} \sqcap \mathsf{B})$) unless we explicitly state that $r : \neg B$. To solve this issue, we simplified the mapping to represent <wsp:exactlyOne> as logical disjunction (inclusive OR), and in addition we have made the classes representing the alternatives pair-wise disjoint, so even though a requester supports more than one alternative, he cannot use more than one at a time. This updated translation is more concise than the old one (compare $A \sqcup B$ with $(A \sqcup B) \sqcap \neg(A \sqcap B)$). In this scenario, if a requester comes in that is a member of two alternatives, we will get an inconsistency.

Example 1. Consider the example policy in Figure 1. For each policy assertion, we have a separate OWL class (`RequireDerivedKeys`, `WssUsernameToken10`, `WssUsernameToken11`). Then, each alternative is simply the conjunction of its assertions.

$$\mathtt{Alt}_1 \equiv \mathtt{RequireDerivedKeys} \sqcap \mathtt{WssUsernameToken10}$$
$$\mathtt{Alt}_2 \equiv \mathtt{RequireDerivedKeys} \sqcap \mathtt{WssUsernameToken11}$$

```
<wsp:Policy
   xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy"
   xmlns:wsp="http://www.w3.org/2006/07/ws-policy" >
 <wsp:ExactlyOne>
   <wsp:All>
     <sp:RequireDerivedKeys />
     <sp:WssUsernameToken10 />
   </wsp:All>
   <wsp:All>
     <sp:RequireDerivedKeys />
     <sp:WssUsernameToken11 />
   </wsp:All>
 </wsp:ExactlyOne>
</wsp:Policy>
```

Fig. 1. Example policy

The policy class $\mathtt{P}$ is equivalent to the disjunction of the alternative classes:

$$\mathtt{P} \equiv \mathtt{Alt}_1 \sqcup \mathtt{Alt}_2$$

In addition, we add a disjoint axiom for the alternatives:

$$\mathtt{Alt}_1 \sqsubseteq \neg\mathtt{Alt}_2.$$

4 Policy Processing Services

In our previous work [13] on WS-Policy, we described the services that DL reasoners provide regarding policies: containment, equivalence, incompatibility, incoherence (nothing can satisfy the policy) and policy conformance, among others. Thus, the mapping allows us to use an off-the-shelf OWL reasoner as a policy engine and analysis tool, and an off-the-shelf OWL editor as a policy development and integration environment. OWL editors can also be used to develop domain specific assertion languages (essentially, domain ontologies) with a uniform syntax and well specified semantics.

There is one additional reasoning service that is useful for policies and warrants more discussion. It has been argued (see [4] for example) that explanation is a crucial requirement for a policy language. To address this requirement, we can use recent advances in the field of debugging OWL ontologies [11], esp. in providing explanations for both ontology inconsistencies and arbitrary entailments for OWL-DL.

For example, the *why* query mentioned in [4] can be handled by the explanation for arbitrary entailments. If a user asks why the requester $\mathtt{r}$ satisfies the policy $\mathtt{P}$, then the debugging framework is simply asked to provide justification for the type assertion $\mathtt{r}$:$\mathtt{P}$. On the other hand, if a web service request causes an inconsistency (for example because of violating a domain disjointness constraint), then the debugging framework can provide explanation of why the inconsistency occurred. More specifically, if an OWL-DL ontology is inconsistent, [11] provides the minimal set of axioms in the ontology that causes the inconsistency (the set of axioms is called a *justification*).

These techniques are already implemented in Pellet, and there is also a UI for debugging implemented in SWOOP.

5 Policy Intersection

Policy intersection is used when a web service requester and provider both express policies and want to compute the compatible policy alternatives between them. This commutative and associative function takes two policies as input and returns a policy containing the compatible alternatives. As defined in [21], two alternatives are compatible if each assertion in the first alternative is compatible with an assertion in the second, and vice-versa. If two policy alternatives are compatible, their intersection is an alternative containing all of the assertions in both alternatives.

Determining whether two policy alternatives are compatible involves domain-specific processing. In an attempt to automate the operation, one might be tempted to mark the incompatible policy assertions as mutually disjoint classes. Then, to determine whether two policies A and B are compatible we only check whether $\mathsf{A} \sqcap \mathsf{B}$ is satisfiable. However, this will prevent us from having entities support assertions of different types, since it will render the policy ontology inconsistent. Since it is usually the case that entities do support different assertion types (example: an entity can support some specific encoding and some type of reliability, and encoding and reliability are different assertion types), the simple approach of marking incompatible assertions as disjoint classes is incorrect.

To overcome this problem, we introduce an additional property in the policy ontology - `compatibleWith`. Then, for two policy assertion classes A and B, if we want to say that A is not compatible with B, we can simply use $\mathsf{A} \sqsubseteq \neg\exists \texttt{compatibleWith}.\mathsf{B}$.

As stated in [21], assertion authors are encouraged to factor assertions such that two assertions of the same assertion type are typically compatible. We can model this using inheritance hierarchies (with exceptions). For instance, the policy modeler can state that for two classes representing assertions C, D, which she knows are compatible, every pair of classes C_i, D_i that are subclasses of C, D (i.e., $C_i \sqsubseteq C$ and $D_i \sqsubseteq D$) is also compatible by default. This can be expressed with the following default rule:

$$\frac{\mathsf{C}(x) \wedge \mathsf{D}(y)\ :\ \texttt{compatibleWith}(x, y)}{\texttt{compatibleWith}(x, y)}$$

In the cases when two assertions are incompatible (even though they are a inherit from the same type) the policy developer can add a disjoint axiom by hand, overriding the default rule above.

The basic algorithm would be as follows: for two policies A and B and a default theory $KB = \langle \mathcal{W}, \mathcal{D} \rangle$ (where $\mathcal{W}$ is an OWL-DL ontology and $\mathcal{D}$ is a set of defaults), to determine whether they are compatible start with the alternatives of A and try to find one compatible alternative in B, and vice-versa. If for at least one alternative in one policy, we succeed in finding compatible alternatives in the other policy, we conclude that the policies can intersect. The intersection of the policies is the policy containing the mutually compatible set of alternatives. To determine whether two alternatives are compatible, we try to match their assertions. For each assertion $\texttt{Assert}_{\texttt{a}} \in \mathsf{A}$, we try to find an assertion $\texttt{Assert}_{\texttt{b}} \in \texttt{B}$ s.t. $KB \models \texttt{compatibleWith}(\texttt{Assert}_{\texttt{a}}, \texttt{Assert}_{\texttt{b}})$. If the assertion has a nested policy, then we try to match it with a nested policy from the other alternative, by asking recursively whether they are compatible.

6 OWL Defaults

Both of the default logic scenarios described above could be plausibly met with Reiter's default logic, which is one of the most studied non-monotonic logics. Reiter's default logic, while very expressive, is, like many non-monotonic formalisms, known to be computationally difficult even in the propositional case. In [3], Baader and Hollunder showed that even a restricted form of defaults coupled with a description logic that contains a smaller set of constructors than OWL-DL was undecidable. They also showed that if one restricted the defaults to apply only to named individuals (or, equivalently, restricted the logic to closed defaults), then a robust decidability ensued.

We have implemented a prototype of the terminological defaults of Baader and Hollunder that is based on recent advances in description logic reasoning: tableaux tracing for the description logic $\mathcal{SHOIN}$ and incremental reasoning support. The implementation is provided as an extension to Pellet and it provides *realization* of individuals in terminological default theories. We have also provided a UI for defaults by extending the open source OWL Ontology editor SWOOP. More specifically, we added support for default rules editing and updating the current ontology with the set of inferred facts from the defaults. We refer the reader to [12] for more details.

7 Related Work

There have been a number of proposals for ontology-based web policy systems [16, 10, 18, 8] - because of lack of space, we will only briefly cover Rei and KaOS.

Rei [10] is a policy specification language based on a combination of OWL-Lite, logic-like variables and rules. It allows users to develop declarative policies over domain specific ontologies in RDF and OWL. Rei allows policies to be specified as constraints over allowable and obligated actions on resources in the environment. A distinguishing feature of Rei is that it includes specifications for speech acts for remote policy management and policy analysis specifications like what-if analysis and use-case management. Our goal is to encode WS-Policy in a not very expressive logic formalism (so as to be able to perform policy analysis), and our opinion is that we do not need a language as expressive as Rei for WS-Policy.

KaOS Policy and Domain Services [18] use ontology concepts encoded in OWL to build policies. These policies constrain allowable actions performed by actors which might be clients or agents. The KAoS Policy Service distinguishes between authorizations and obligations. The applicability of the policy is defined by a class of situations which definition can contain components specifying required history, state and currently undertaken action. Even though we use the same representation language as KaOS (OWL-DL), our reasoning support is provided by tableaux-based description logic reasoners which are sound and complete for OWL-DL. In addition, by using Pellet we were able to leverage its ontology debugging support.

In addition, there are a number of proposals [20, 14] of policy/authorization languages based on logic programs extended with default rules - the difference with our approach is that we use description logics as the underlying logic formalism.

8 Conclusions and Future Work

While most policy language proposals are based on logic programs, in this paper we explored the alternative of using OWL-DL as a language for expressing web service policies. We argued that the policy services that DL reasoners provide out of the box, the advances in explanation mechanisms for DL, and the ability to closely integrate OWL-DL with default logic make an OWL-based policy framework worth exploring. Also, OWL-DL is a W3C standard, a language with clear syntax and semantics that is ubiquitous in the Semantic Web. As a consequence, the number of reasoners and OWL-DL editors has been growing steadily. A policy language based on OWL-DL should be able to capitalize on the popularity of OWL-DL.

Despite the advantages mentioned above, policies, being associated with rules first and foremost, seem to demand greater expressivity than OWL-DL (as argued in [9], for example) in the form of monotonic rules. However, because of the recent advances in hybrid (description logic + logic programs) knowledge bases, and successful implementations ([15]) we believe that OWL-DL combined with rules is reaching a maturity level where it will be a suitable alternative for a policy framework.

During the past couple of years, there has been great advances [7, 5, 19, 17] in the area of automated trust negotiation (ATN) between policy entities. ATN deals with the problem of exchanging of sensitive credentials between strangers in order to establish trust. We plan to investigate how we can integrate our OWL-based system with such mechanisms.

Finally, it is unfortunate that we cannot provide clear semantics for policy intersection because its dependence on domain-specific reasoning. The WS-Policy framework requires each domain to specify its own policy assertions, but there is no generic, domain-independent language for expressing these assertions. As a result, every domain has its own language (with unclear semantics) that makes it hard to reason and analyze the assertions. We plan to investigate how we could couple OWL with concrete domains (e.g. XPath) so as to be able to express and give semantics to some of these domains. A promising step toward a domain-independent policy assertions language is [2]; we plan to investigate the idea further.

References

1. A. H. Anderson. An introduction to the web services policy language. In *Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04)*, 2004.
2. Anne Anderson. WS-PolicyConstraints: A domain-independent web services policy assertion language, November 2005. Available at http://research.sun.com/projects/xacml/IntroToWSPolicyConstraints.pdf.
3. Franz Baader and Bernhard Hollunder. Embedding Defaults Into Terminological Knowledge Representation Formalisms. *J. Autom. Reasoning*, 14(1):149–180, 1995.

4. P.A. Bonatti, G. Antoniou, M. Baldoni, C. Baroglio, C. Duma, N. Fuchs, A. Martelli, W. Nejdl, D. Olmedilla, J. Peer, V. Patti, and N. Shamheri. The rewerse view on policies.
5. Piero A. Bonatti and Pierangela Samarati. A uniform framework for regulating service access and information release on the web. *J. Comput. Secur.*, 10(3):241–271, 2002.
6. Jacques Durand et al. Wsdl annotation proposal. http://lists.oasis-open.org/archives/wsrm/200403/msg00082.html.
7. R. Gavriloaie, W. Nejdl, D. Olmedilla, K. Seamons, and M.Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In *European Semantic Web Symposium*, May 2004.
8. Stephan Grimm, Steffen Lamparter, Andreas Abecker, Sudhir Agarwal, and Andreas Eberhart. Ontology based specification of web service policies. In *Semantic Web Services and Dynamic Networks Workshop*, 2004.
9. R. Montanari J. Bradshaw, L. Kagal and A. Toninelli. Rule-based and ontology-based policies: Toward a hybrid approach to control agents in pervasive environments. In *Proceedings of the ISWC2005 Semantic Web and Policy Workshop*, 2005.
10. L. et al Kagal. A policy language for a pervasive computing environment. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, June 2003.
11. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging unsatisfiable classes in owl ontologies. *Journal of Web Semantics - Special Issue of the Semantic Web Track of WWW2005*, 3(4), 2005.
12. Vladimir Kolovski, Bijan Parsia, and Yarden Katz. Implementing owl defaults. Technical report, University of Maryland - College Park, 2006. http://www.mindswap.org/ kolovski/defaults.pdf.
13. Vladimir Kolovski, Bijan Parsia, Yarden Katz, and Jim Hendler. Representing web service policies in owl-dl. In *International Semantic Web Conference (ISWC)*, 2005.
14. Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation Logic: A logic-based approach to distributed authorization. *ACM Transaction on Information and System Security (TISSEC)*, February 2003.
15. Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for owl-dl with rules. In *Proc. of ISWC 2004*, pages 549–563.
16. W. Nejdl, D. Olmedilla, M. Winslett, and C. Zhang. Ontology-based policy specification and management. In *2nd European Semantic Web Conference (ESWC)*, May 2005.
17. K. Seamons, M. Winslett, and T. Yu. Limiting the disclosure of access control policies during automated trust negotiation, 2001.
18. A. Uszokand and J. Bradshaw. Kaos policies for web services. In *W3C Workshop on Constraints and Capabilities for Web Servies*, October 2004.
19. W. Winsborough, K. Seamons, and V. Jones. Automated trust negotiation. Technical Report TR-2000-05, 24 2000.
20. T. Y. C. Woo and S. S Lam. Authorization in distributed systems : A formal approach. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 33–51, 1992.
21. WS-Policy. Web services policy framework (ws-policy). http://www-106.ibm.com/developerworks/library/specification/ws-polfram/.

Semantics in Model-Driven Business Design

Mark H. Linehan

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
mlinehan@us.ibm.com

Abstract. This position paper describes ongoing work in applying the new OMG standard called *Semantics in Business Vocabulary and Rules* (SBVR) to a model-based approach to business design and implementation. The work explores methods of specifying semantics and rules in SBVR's "Structured English" as extensions of business models that are automatically translated to executable solutions.

Introduction

The Object Modeling Group's (OMG's) Model-Driven Architecture [13] concept defines a multi-layered approach to defining business solutions, as shown in figure 1.

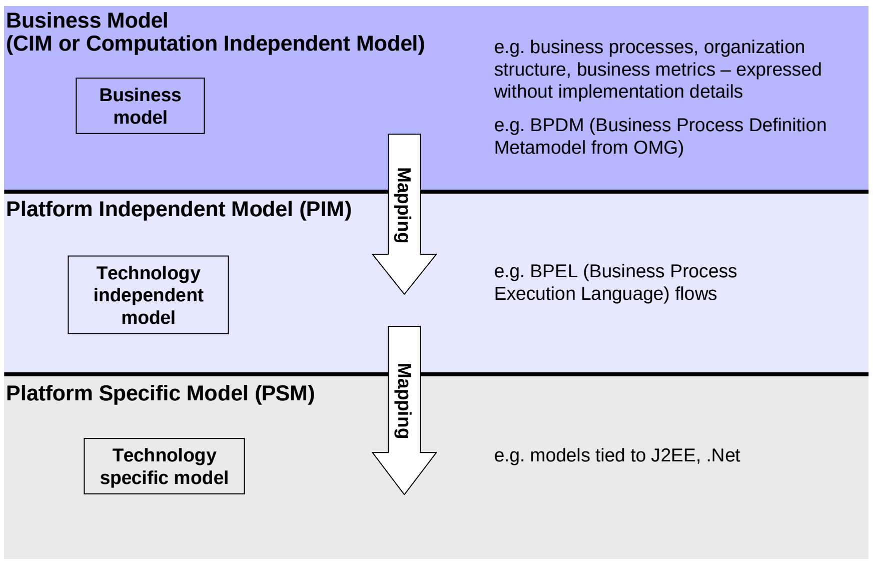

Business Model
(CIM or Computation Independent Model)
Business model
e.g. business processes, organization structure, business metrics – expressed without implementation details
e.g. BPDM (Business Process Definition Metamodel from OMG)
Mapping
Platform Independent Model (PIM)
Technology independent model
e.g. BPEL (Business Process Execution Language) flows
Mapping
Platform Specific Model (PSM)
Technology specific model
e.g. models tied to J2EE, .Net

Fig. 1. OMG Modeling Layers

Figure 2 summarizes OMG efforts to define standards for rules at the top two layers. The *Semantics of Business Vocabulary and Rules* [15] activity is defining a "Structured English" approach to vocabulary and rules at the Business Model or Computation Independent Model layer. The *Production Rules Representation* [14] aims to specify a standard Unified Modeling Language (UML) model for rule structures.

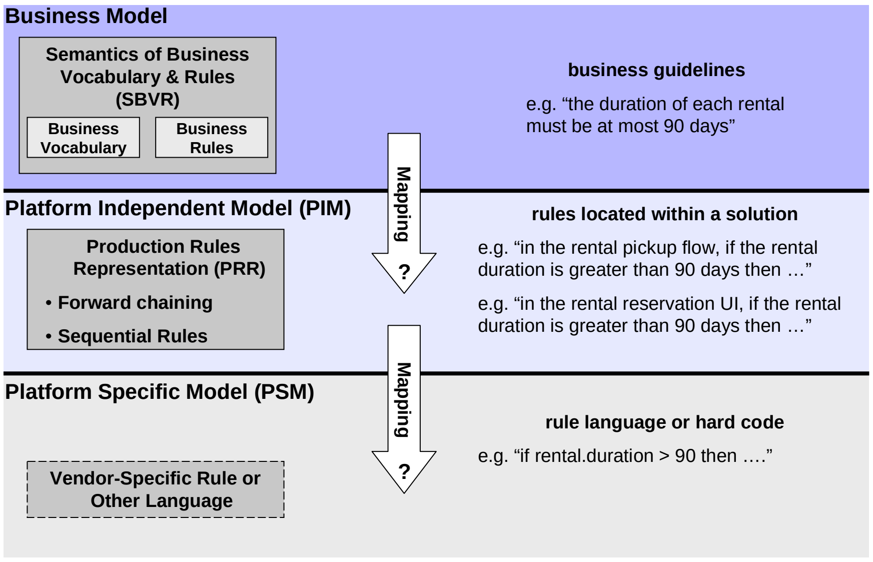

Business Model
Semantics of Business Vocabulary & Rules (SBVR)
Business Vocabulary
Business Rules
business guidelines
e.g. "the duration of each rental must be at most 90 days"
Mapping
?
Platform Independent Model (PIM)
Production Rules Representation (PRR)
• Forward chaining
• Sequential Rules
rules located within a solution
e.g. "in the rental pickup flow, if the rental duration is greater than 90 days then …"
e.g. "in the rental reservation UI, if the rental duration is greater than 90 days then …"
Mapping
?
Platform Specific Model (PSM)
Vendor-Specific Rule or Other Language
rule language or hard code
e.g. "if rental.duration > 90 then …."

Fig. 2. Rules in the MDA Layers

This paper summarizes an ongoing effort to implement a subset of SBVR in the context of an existing *Model Driven Business Transformation* project [10] at IBM Research.

MDBT – Model-Driven Business Transformation

MDBT is a methodology and matching toolkit for defining a business solution at the business modeling layer, and then semi-automatically transforming the solution into a PIM-layer and then a PSM-layer implementation. A business analyst applies the methodology by defining a business model using the IBM WebSphere Business Modeler [7] tool and the MDBT semantics. The analyst then converts the business model to a PIM-layer model using the IBM Rational Software Architect [6] product, and further transforms the PIM-layer model to an executable implementation using the IBM WebSphere Integration Developer [8] tool and IBM WebSphere Process Server [9] runtime. The generated implementation includes Data Definition Language (DDL) statements to generate relational database tables, state machine definitions for executing the solution, skeleton user interface Java Server Pages (JSPs), and service definitions in the form of Web Services Definition Language (WSDL) files. The implementation incorporates business performance monitoring functions and dashboards, as described in [2].

The transformation process from business layer to implementation can be fully automated in a "rapid prototyping" mode. Manual intervention at the PIM and PSM layers are needed to produce production-quality user interfaces and adapters for invoking legacy systems as services.

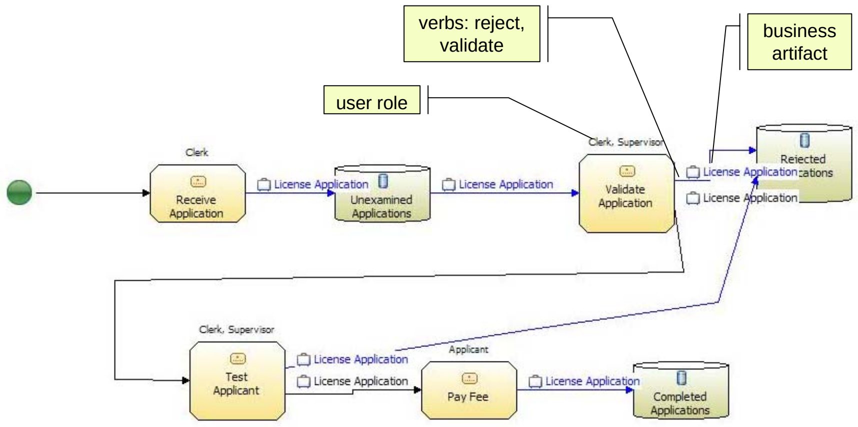
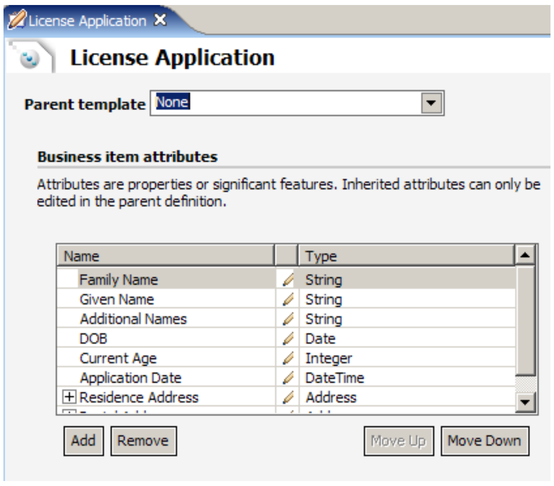


Fig. 3. Business-Layer Model of a Driver's License Bureau

Figure 3 shows a simple solution example at the business layer. This shows the processing flow of a Driver's License Bureau which handles License Applications. The flow starts at the dot on the left, and proceeds through the illustrated stages. The rounded squares show processing tasks, while the database icons show repositories for holding in-process work. The rectangular callouts indicate the three primary concepts captured in this model: user roles, verbs associated with the output sides of tasks, and the business artifacts processed by the solution.



License Application

Parent template: None

Business item attributes

Attributes are properties or significant features. Inherited attributes can only be edited in the parent definition.

Name	Type
Family Name	String
Given Name	String
Additional Names	String
DOB	Date
Current Age	Integer
Application Date	DateTime
Residence Address	Address

Fig. 4. Business Artifact

At the business layer, artifacts are detailed in terms of their attributes. Figure 4 shows that a License Application contains various fields, similar to properties in UML classes.

What's missing from the business-layer model is any concept of business rules. For example, perhaps the applicant must be at least 18 years old to get a driver's license. In the current MDBT approach, such rules must be implemented manually at the PSM layer. A method of specifying such rules at the business layer and then transforming them to the implementation would improve the MDBT methodology. The objective of this project is to examine the suitability of SBVR for this purpose.

SBVR

SBVR provides a framework for defining business vocabulary and rules at the business modeling layer using "Structured English" and applying stylized text to four key concepts:

– The 'term' style applies to noun concepts, such as 'License Application'.
– The 'Name' style designates individual concepts, such as a clerk named 'Bill'.
– The '*verb*' style identifies fact types, which define relationships between concepts. For example, 'clerk *validates* application'.
– The 'keyword' style distinguishes various words used to construct vocabulary definitions and rule statements. The keywords designate built-in SBVR concepts such as 'it is permitted that' and 'exactly one'.

SBVR supports standard logical operations ('and', 'or', and so forth) and first order predicate logic (e.g. 'each', 'some'). SBVR also supports certain modal logic concepts such as necessity, possibility, obligation, and permission. Some example rules given in "Structured English" are:

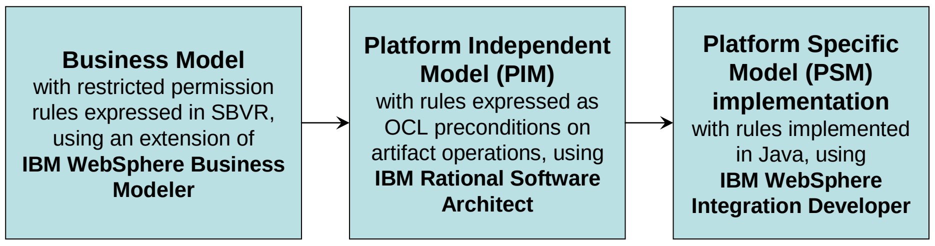
It is permitted that each clerk *validates* each license application only if the current age *of* the license application *is greater than* 18.

It is obligatory that each applicant *pass* the written test.

Note the influence of the vocabulary design on the expression of the rules. The first example references the "the current age *of* the license application," rather than "…*of* the applicant," because "current age" is a field of the "license application" artifact. The vocabulary – and perhaps the underlying application – would have to be restructured to enable a more natural rule statement.

The project described here is creating a prototype tool to evaluate the technical issues involved in writing SBVR rules, and then transforming them to executable implementations.

Prototype Design



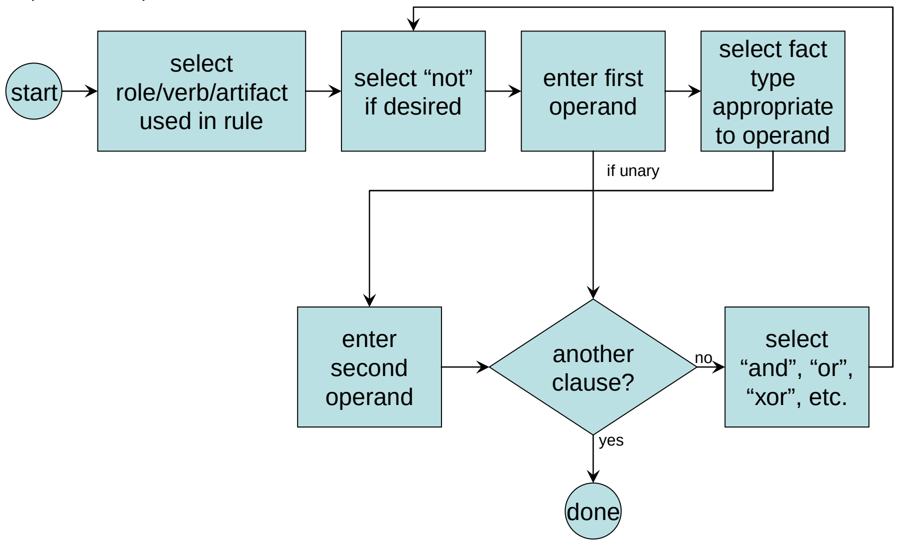
Business Model
with restricted permission rules expressed in SBVR, using an extension of
IBM WebSphere Business Modeler
Platform Independent Model (PIM)
with rules expressed as OCL preconditions on artifact operations, using
IBM Rational Software Architect
Platform Specific Model (PSM) implementation
with rules implemented in Java, using
IBM WebSphere Integration Developer

Fig. 5. Prototype Summary

The SBVR prototype is designed as an extension of the MDBT project, in order to build upon the existing MDBT modeling and transformation technology. As shown in figure 5, the rules are entered in a new tool added onto the WebSphere Business Modeler, and then transformed into a PIM-layer solution, and then further transformed into a PSM-level implementation.

Specifying Rules in the Business Model

The SBVR specification is large and fairly complex. Rather than attempt to support the entire specification, this prototype focuses on a limited subset called "restricted permission rules". These are rules expressed as permissions (someone or something *may* do something) associated with conditions. The first example rule given above is a restricted permission rule. In the prototype, all such rules are associated with an MDBT business model such as the one shown above. Each rule references a user role, an action, and a business artifact in the model.



start
select role/verb/artifact used in rule
select "not" if desired
enter first operand
select fact type appropriate to operand
if unary
enter second operand
another clause?
no
select "and", "or", "xor", etc.
yes
done

Fig. 6. Rule Wizard Navigation Path

Two methods of entering rules could be imagined. One involves a tool that parses "Structured English" text and attempts to discover the underling meaning. The problem with that approach is that all text – even "Structured English" – has ambiguities. Manual user involvement would be required to resolve those ambiguities.

This prototype employs an alternate approach in which rules are entered through a tool "wizard" that guides users, step-by-step, through the process of creating a complete rule. Figure 6 summarizes the wizard steps. Advantages of this approach are two-fold: (a) users can only enter valid rules; (b) the meaning of the rules is explicit.

Transforming from Business Model to Platform Independent (PIM) Model

Following the MDBT technology, the rules entered through the wizard are converted to a PIM-layer solution as part of the overall MDBT transformation. MDBT models the PIM layer using UML class, state, and use case diagrams. This prototype extends the class diagrams by converting the rules to pre-conditions on the class operations. These pre-conditions are expressed using the Object Constraint Language [17].

Advantages of OCL for this purpose include the fact that it is an established standard, the potential to convert to any appropriate PIM-layer implementation, and OCL's built-in collection operators. The latter facilitate SBVR's use of universal and existential logic. For example, a rule fragment such as "each line item *of* the order *is complete*" may be converted to an OCL fragment such as "lineItems→select(isComplete)".

Transforming from Platform Independent Model to Platform Specific (PSM) Model

The transformation from PIM-layer to PSM-layer potentially converts the pre-conditions to equivalent tests in various aspects of the implementation. These include code that enables or disables buttons in the user interface, guards on state machine transitions, and potentially access control statements in a language such as the eXtensible Access Control Markup Language (XACML) [12]. The first example rule given above might map to all of these. This illustrates the power of SBVR and the MDBT approach: a single rule given at the business layer potentially drives multiple aspects of the ultimate solution.

For simplicity, this prototype converts the OCL preconditions only to Java. The transformation is simple, except that collection operators must be generated as corresponding "for" loops. Mappings to implementations such as rule languages, XACML, script languages, and others are possible and relatively easy with the MDBT approach.

Related Work

Since SBVR is quite new, relatively little work has been published about it. One announced commercial implementation and a research prototype of SBVR exist:

- RuleExpress is a commercial SBVR tool produced by a collaboration of Business Rule Solutions, LLC [1] and LibRt [11]. The website [16] says RuleExpress provides “Business-people capabilities for business rules … capture, expression, validation, verification, visualization, management, publication, audit.”
- SBeaVer [3] is an open-source SBVR tool created by Maurizio De Tommasi and Pierpaolo Cira at the University of Lecce in Italy, in a project funded by the European Digital Business Ecosystem [4] project. This tool runs as a plugin for the Eclipse [5] tools platform, and provides for creation, editing, validation, verification, and export of both vocabulary and rules.

These tools enable entry and modification of rules and vocabulary using “Structured English”. In contrast, the work described here focuses upon transformation of the rules to executable code.

Summary and Outlook

The project experience so far is that entering rules in SBVR “Structured English” seems to be a useful adjunct of the existing Model-Driven Business Transformation (MDBT) technology. Conversion of the limited subset of SBVR supported by this prototype into OCL and Java is straightforward.

This prototype addresses a small portion of the concepts defined by SBVR. Features of particular interest for future work include:

- Synonyms to permit alternative terms and part of speech in rules. For example, a ‘License Application’ might also be named an ‘Application’.
- Noun and fact type definitions to simplify the expression of certain rules. For example, rather than specifying rules for when a clerk may validate an application, one might define a ‘valid application’ according to a set of conditions.
- Additional modalities, such as necessities and obligations. The second example above gives an example of an obligation rule.

These three require very different kinds of technology. Synonyms are entirely a matter of tools function. Definitions and other modalities require either new transformations among modeling levels or new execution mechanisms such as inferencing.

SBVR brings together concepts from several distinct traditional academic subjects: philosophy (modal logics, taxonomies), linguistics (semantics, pragmatics), and mathematics (first order logic). The application of these concepts to computer science topics such as modeling, model transformation, Description Logics, and rules, offers rich opportunities for scientific and technical progress. The expression of these concepts in "Structured English" promises to make rules practical and useful for everyday business solutions.

References

1. Business Rule Solutions, LLC. See http://www.brsolutions.com/.
2. Chowdhary, P., Bhaskaran, K., Caswell, N. S., Chang, H., Chao, T., Chen, S.-K., Dikun, M., Lei, H., Jeng, J.-J., Kapoor, S., Lang, C. A., Mihaila, G., Stanoi, I., Zeng, L. "Model Driven Development for Business Performance Management." *IBM Systems Journal*, Volume 45, Number 3, Page 587 (2006). Available at http://www.research.ibm.com/journal/sj45-3.html
3. De Tommasi, Maurizio, Cira, Pierpaolo. SbeaVer Business Modeler Editor. Available at http://sbeaver.sourceforge.net.
4. Digital Business Ecosystem project, "an Internet-based software environment in which business applications can be developed and used". Available at http://www.digital-ecosystem.org/.
5. "Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software." Available at http://www.eclipse.org/.
6. IBM Rational Software Architect. See http://www-306.ibm.com/software/awdtools/architect/swarchitect/index.html.
7. IBM WebSphere Business Modeler. See http://www-306.ibm.com/software/integration/wbimodeler/.
8. IBM WebSphere Integration Developer. See http://www-306.ibm.com/software/integration/wid/.
9. IBM WebSphere Process Server. See http://www-306.ibm.com/software/integration/wps/.
10. Koehler, Jana; Hauser, Rainer; Kapoor, Shubir; Wu, Fred Y.; and Kumaran, Santhosh. *A Model-Driven Transformation Method*. In Proceedings of the Seventh International Conference on Enterprise Distributed Object Computing, pages 186--197. IEEE, September 2003. Available at http://doi.ieeecomputersociety.org/10.1109/EDOC.2003.1233848.
11. LibRT. See http://www.librt.com/.
12. Organization for the Advancement of Structured Information Standards (OASIS). *eXtensible Access Control Markup Language* (XACML). See http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.
13. Object Modeling Group (OMG). *MDA Guide*, version 1.01, 2003. Available at http://www.omg.org/docs/omg/03-06-01.pdf.
14. Object Modeling Group (OMG). *Production Rules Representation Revised Submission*, June 5, 2006.
15. Object Modeling Group (OMG). *Semantics of Business Vocabulary and Rules Specification Drafted Adopted Specfication*, March 2, 2006.
16. RuleExpress, "The business tool for expressing and communicating business rules." Available at http://www.rulexpress.com/index.php.
17. Warmer, Jos, Kleppe, Anneke. *The Object Constraint Language: Getting Your Models Ready for MDA*. second edition, Addison-Wesley Professional; 2003; ISBN 0321179366.

Non-Boolean Authentication

Alec Yasinsac

Florida State University, Computer Science Department
Tallahassee, Florida 32306-4530 USA

Abstract. Traditional authentication is two valued. Unfortunately, authentication mechanisms cannot perfectly establish electronic participant's identity. Despite years of research and its manifestations such as digital signatures, zero knowledge proofs, public key infrastructures, certificates, biometric tools, etc. the best authentication evidence is a combination of multiple factors. All authentication systems are imprecise, but there are no existing systems that capture or that facilitate reasoning about this property. This paper introduces many fundamental issues in multi-tiered authentication systems.

1 Introduction and Motivation

In theory, authentication is Boolean; either someone is who they say they are, or they are not. Unfortunately, as any good practioner will tell you: "In theory, theory and practice are the same, but in practice, they are not". Unfortunately for information security, this "practicality axiom" holds true with authentication; that is, in general it is practically impossible to establish absolute authentication. Sophisticated intruders can guess, mine, or acquire passwords through social engineering. Private keys can be stolen or (more likely) mishandled. Adversaries can electronically capture biometric information or compromise underlying biometric security protocols.

Still, most trust systems treat authentication as though it were Boolean. Even in systems that partition trust into levels [1] there are few approaches (if any) that can cope with varying *authentication confidence* levels.

We introduce a model, architecture, and mechanisms that accommodate the reality that authentication is rarely Boolean. We rely on abstract notions of limited transitive trust with time-sensitive, information maturity and growth in a multi-level authentication model. Our architecture is a two-tiered structure that allows action categories that active responses offset as additional authentication information emerges. Our mechanisms focus on independent, cooperating identity sensors and state reversion.

1.1 Multi-State Authentication

Security systems canonically have two authentication states, roughly corresponding to (1) Identity Authenticated and (2) Identity Not Authenticated. Until we properly enter our account identifier and password, we are "not authenticated", so we receive no access privileges. We are so accustomed to this paradigm that it may be hard to imagine how an n-tiered authentication confidence scheme may work. Let us illustrate.

Most of us have experienced account suspense as a result of failing to correctly enter our password in three attempts. Account suspense after three failed authentication tries is one common practice that recognizes a third authentication class, call it *Identity Claim Disproven* (ICD). Essentially, the ICD authentication category reflects a negated identity claim or that a mechanism verified that a false identity claim occurred. Thus, we identify the following authentication classes within this *three state paradigm*: (1) Identity Unknown, (2) Identity Authenticated, and (3) Identity Claim Disproven.

The three state authentication paradigm leads to numerous research questions, e.g.:

1. Can we systematically categorize authentication confidence states?
2. What are legitimate actions/responses for a given n-state authentication system and how can this state/action relationship be best represented?
3. Can we characterize the optimum, minimum, and maximum number of authentication states for a given protection system?
4. Can we capture the essential authentication properties to allow continuous, incremental re-authentication?

Earlier work [2] investigates possible responses to incomplete authentication based on *vanilla* services. This notion leverages traditional access control and information flow models [3, 4], particularly that different objects have different protection requirements. Intuitively, objects with minimal sensitivity need the minimum or vanilla protection.

A complementary issue relates to proactive responses to incremental authentication and re-authentication. For example, we consider whether or not it is reasonable to reverse actions taken by a partially authenticated party if their identity claim is later refuted or its confidence level downgraded. We offer a general approach that we call *Rollback*.

A fundamental component of this research is to determine if rollback is essential for incremental authentication confidence systems. This idea appears intuitive, i.e. an act made while masquerading should be reversed when the masquerade is discovered. There is little in the literature on systematic approaches to backing-out to a previous secure state, though there is related work concerning disaster recovery that we address in the next section.

1.2 Theoretic Foundations

In their seminal paper, Harrison, Ruzzo, and Ullman introduce mathematical security models for managing computer access control [4]. There are many similar models [1], evaluations [5], and refinements [3] in the literature and research continues [6, 2] with significant interest in access control models for ubiquitous computing [7, 8]. Different environments demand different security models, and computing continues to change at breakneck pace. Access control models are not keeping pace with this change.

The literature is also rich with works targeting authentication definition [9,10] and properties [11] with an early, extended bibliography in [12]. Most recent work focuses on cryptographic authentication techniques triggered by [13], with seminal works by Burrows, et al.[14], Lampson et al. [15], Diffie, et al. [16], and Bird et al. [17] with a litany of variations [18, 19, and many others].

A common thread of this work is that it distinguishes only two authentication states. Work in threshold cryptography [20] offers an environment that has inherent opportunity for multi-state authentication and response, but we have seen no such work in the literature. We examine the opportunity in this area in this paper.

2 Multi-tiered Authentication Confidence States

2.1 Foundations in the Three State Model

We begin this description by adopting the three-tiered *three state model*, as described earlier, as our foundation. We fix the endpoints at "perfect confidence" with the

Identity Authenticated state on one flank and *Identity Claim Disproven* (ICD) on the other. ICD users are denied all access while access for fully identity authenticated users are controlled by the normal access control system. Our primary interest lies in the middle state: *Identity Unknown*.

We consider the three level model foundational because here we prove and exercise the concept of vanilla access that is granted to *Identity Unknown* subjects. The term "vanilla" seems particularly applicable as an intentional double-entendre. First, it reflects a plainness that characterizes the least protection afforded objects in a protection system. Vanilla objects require no special access control because they are not sensitive, either for confidentiality, integrity, or availability. Since they require [essentially] no protection, unknown subjects may access them. Depending on the environment, there may be a rich set of vanilla services, or there may not be any.

2.2 Vanilla Users

The vanilla user notion is evident in a variety of open laboratory environments. For example, many university libraries do not require user authentication on library computers. In some cases, the only applications available on accessible terminals provide library search capabilities. In general, such library search applications are not sensitive; in fact library patrons are encouraged to utilize these systems to locate resources without engaging reference personnel. We might call this system, vanilla-only access or a single state model.

A mild adjustment to the library illustration of requiring authentication for system administrators using library computers reflects the earlier described two-state model. In this scenario, an authentication system partitions users into the *identity unknown* and *identity authenticated* classes. Once authenticated, administrators have special access privileges not available to vanilla (unknown) users. A central theme of our paper is that access states may be monotonic, e.g. administrators are inherently vanilla users and need not be authenticated to receive vanilla access.

To extend the library illustration to a three state model, we require weak authentication for all users. For example, the authentication may be so simple as swiping a student identification card or entering a library issued group key, reflecting the likely status of the user being a university student. The classes in the illustration are:

(1)	[Specific] Identity Unknown:	Vanilla university students
(2)	Identity Authenticated:	System administrators
(3)	Identity Claim Disproven:	Users failing student authentication

In this simple illustration, system responses for vanilla users seem reasonably clear. They may access any provided library applications as often as they like, for as long as they like. If the applications allow file writing, the user may write to the files through applications. Of course, some libraries may set more liberal or more restrictive access policies for vanilla users, but these seem to reflect vanilla access for this illustration.

The more interesting question relates to limitations on vanilla users. Clearly, they are not allowed to perform system administration functions, such as installing programs or editing existing program or system configuration. Possibly not so clear is whether other general, non-sensitive functions (such as web browsing, Internet chat, even simple file editing, say through Notepad) are available. In the three state model, the system owner must decide if any of these applications should be available on the library nodes, and if they should be available only to system administrators or to all vanilla users.

2.3 The N-State Model

The core of this paper is to partition the vanilla state to form an n-state model, where n is greater than three, e.g. Figure 1. We begin by describing a state split to form a four state model, and then give a theory regarding further partitioning and refinement. Central to this process is how we identify vanilla session classes that correspond to vanilla object classe, and reasonable respective responses.

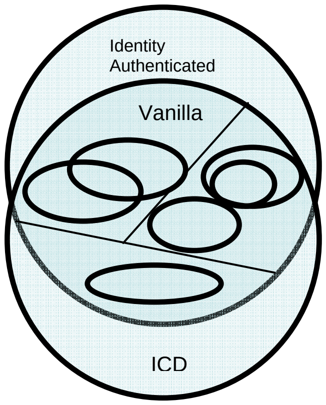


Figure 1

2.3.1 Incremental Session Re-authentication

Many security models (e.g. [1]) are founded on the notion of tranquility, that is, that subjects and objects' security posture does not change. Conversely, a foundation of this paradigm is that while objects are tranquil, the authentication posture of each subject in every session may continuously change. For most cases, we expect to gain authentication confidence with time, eventually reaching the *identity authenticated* state and remaining in that state with access controlled by the normal protection system.

Conversely, we contend that re-authentication should be continuous as, e.g.:

(1) An authentic user is unable to successfully complete the authentication process
(2) An intruder advances into a vanilla authentication state
(3) A session involving an authenticated, or partially authenticated, user is hijacked by an intruder

While these are three distinct situations, each can be resolved by invoking a continuous authentication process along with a dynamic access control mechanism. Many identity indicators support continuous inspection and incremental reevaluation.

1. Personal Entropy. Beyond biometric mechanisms that may comprise normal authentication systems, humans have characteristic, involuntary behavior that can uniquely identify them. Keystroke pattern (made famous during Carnivore [21] discussions) is one such behavior.

2. Functional behavior. Humans are creatures of habit, thus form behavior patterns that identify them as distinctly as physical and biological characteristics. Intrusion detection systems adopted behavioral profiling as early as 1986 [22].

3. Password hamming Distance. One of the most common authentication errors is the mis-typed password. Present password protection approaches are designed to prevent, rather than leverage, password similarity analysis. We examine mathematical metrics to password protection measure password accuracy.

4. Stored semi-private information. A common authentication approach is to store semi-private user information. Items such as birthday, mother's maiden name, etc. are public information, thus are not strong authentication. In combination with other mechanisms, they provide corroboration that is the essence of vanilla access control.

5. Peer confirmation. Though not fool-proof, personal identification is one of the most reliable authentication mechanisms.

6. Threshold schemes. Threshold schemes [20] partition a secret (e.g. that proves identity) and distribute the shares to several different share-holders. In this paper, we investigate threshold mechanisms that recognize the number of accumulated signatures.

Incremental identification allows vanilla user partitioning so that object access can receive appropriate protection in an unsure world. We make a simple extension, this time of the three state model, to generate a four state model. For example, we may categorize a session as *strong vanilla* if the user entered (1) A correct account identifier (2) An entry that differed from the correct password by a hamming distance of one, or (3) Both of these entries were accomplished on the first try.

The authentication classes in this *four state model* are:

(1) Vanilla	Access objects in the lowest protection level
(2) Strong Vanilla	Users surpassed some, but not all, authentication
(3) Identity Authenticated	Authentication process completed
(4) Identity Claim Disproven	Users whose identity claim is refuted

Classes (1), (3), and (4) are exclusive in the sense that they share no members. Class (2) is a subset of (1). We illustrate these relationships in Figure 2, part a. We then add a fifth class we call *pure vanilla*. We show this class as a subset of strong vanilla in Figure 2 b, but it need not be so. Multiple vanilla classes may form that are proper subsets (as shown in Figure 2), others that are exclusive to one another, and others that overlap, possibly combining all of these architectures within a single protection system.

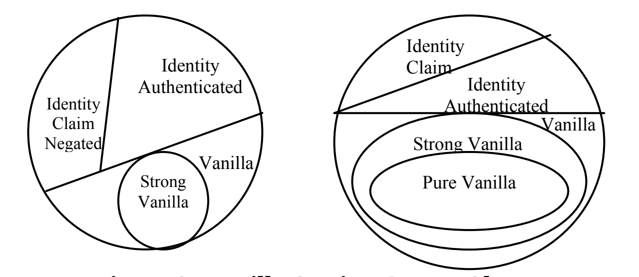


Figure 2. Vanilla Session Access Classes

2.3.2 Classifying Services for Multi-tiered Authentication

We consider how to answer the question of what objects are accessible for a subject-initiated vanilla session. In the three state model, sensitivity is the deciding factor (non-sensitive objects are available to vanilla users). In the four state model, there are two flavors of vanilla sessions, pure vanilla and strong vanilla. We may form corresponding object classes that we may call (1) vanilla and (2) [integrity] sensitive, but recoverable.

The intuition behind this partitioning is that all users whose identity is unsure may access all non-sensitive vanilla data, while users that achieve a threshold of identity confidence may be granted access to sensitive processes as along as the results of those processes are easily reversible (can be rolled back). For example, strong vanilla users may be allowed to add an entry onto the personal calendar associated with its account. These *vanilla* calendar entries are easily removed if the authentication is later refuted.

Notice, we intentionally did not suggest that existing calendar events be revealed to strong vanilla users. The difference is that once revealed information is difficult or impossible to rollback. This does not preclude protection systems from partitioning the vanilla states to allow sensitive information to be revealed to vanilla users, but it is likely that criteria other than rollback potential would guide that permission.

2.3.3 A Mild Formalization

In order to use incremental authentication, we need an implementation structure that supports its semantics. Notionally, we want to be able to add granularity to the access decision. While classification partitions access into sensitivities

Consider a mandatory access control security system consisting of subjects (S), objects (O), classification (C), privileges, and an identity confidence level (ICL), a variation of [1], where classification is a small, ordered, discrete set while the ICL is a continuous vector between zero and one. Subjects and objects are labeled with their classification, which is tranquil. Objects are also labeled with a set of pairs containing a privilege and an ICL, which are also tranquil. When a subject enters the system, they are associated with a dynamic ICL. The security system manages this attribute through mechanisms such as the ones we mention above.

An access request is a triple of the form: AR = {s, o, p}. The access algorithm contains two steps: (1) Decide if the subject and object classifications support granting the desired permission and (2) Ensure that the subject's ICL is high enough to allow the requested action. The former generally follows the Bell-LaPadula structure. We give a simple algorithm for the later in Figure 3. The object icl is extracted from the [classically] static security system object identification file. The subject icl comes from a dynamic record that continuously monitors the subjects' actions and adjusts the icl (again, based on the approaches we mentioned earlier). Security system policy dictates complete mediation, or requires re-authentication when an access durations surpasses some time or volume threshold, this algorithm fully supports the non-tranquility of continuous authentication.

2.3.4 Service Recoverability and Rollback

Previous results [2] identify two situations that allow access privileges to be granted to users that are not fully authenticated. The first is that the information sensitivity does not demand the strongest protection that the security mechanisms provide. The second is whether vanilla privileges actions are reversible, or as we term, can be rolled back.

```
boolean id_confident (s,o,p)
    icl := get_sub_icl(s);
    icl' := get_obj_icl(o,p);
    if icl ≥ icl' return true;
    else return false;
```

Figure 3. ICL Algorithm

The former is mostly a matter of information categorization, similar to that in a multi-level security model such as Bell and LaPadula [1]. An important distinction between Bell and LaPadula and our approach is rollback. Bell-LaPadula-based models assume tranquility because they cannot seamlessly handle down-graded [subject] clearances or upgraded [object] classification. Rollback is one vehicle to offset this dilemma.

Many computer systems and applications require Rollback-type capabilities. Consider file backup systems included in business continuity plans. When important files are lost, properly administered backup systems can return lost files in good working order. File backup issues include currency, immediacy, granularity, history, backup volume capability, and responsiveness, among others. Database recovery systems face similar, though more tightly granular, challenges.

Rollback for security faces many challenges. It is naturally difficult to identify rollback-capable transactions. Clearly, once information is divulged, “forced forgetfulness” is not an option. However, some data items can be easily changed if changed to respond to a dynamic security state. Others cannot be “retracted”.

Similar to other multi-level security models, we must correlate vanilla session state and object vanilla access class. The starting point here is access control matrices and lattice structures, as we illustrate earlier. The novelty lies in the ability to handle dynamic authentication status. Rollback is an essential element. We can also partition confidentiality, integrity [23, 24], and conflict of interest [25] sensitivities where such partitioning facilitates vanilla access capabilities.

3 Conclusion

Authentication has a rich bibliography in theoretical and applied researchfrom some of the top information security researchers in the world. We recognize a reality that is not addressed in previous work, that authentication is not Boolean in practice and that Boolean mechanisms cannot properly characterize security properties in the dynamic Internet and mobile computing environments. This work is particularly relevant to wireless computing environments where peer-to-peer authentication has yet to overcome sophisticated attacks such as Sybil [26] and the invisible node attack [27].

Where absolute authentication is impossible, there must be mechanisms that deal with the uncertain identities. Non-Boolean Authentication enables such mechanisms and offers dynamic multi-level access control designed to leverage (where classical and present operational models prohibit) dynamic privilege assignment and privilege reassignment including classification upgrade and clearance downgrade.

We additionally offer a novel approach to security recovery based on Rollback. Again, we rely on existing work in business continuity planning and database recovery as the foundation for our work. We extend these notions to fit the security perspective and the dynamic authentication environment of worst case attack and Byzantine adversaries.

We base our work on advances that are well-documented in the literature. We leverage lessons learned in security models for confidentiality, integrity, conflict of interest, threshold cryptography, business continuity planning, and many other known technologies to form a comprehensive approach to handle dynamic [re] authentication, classification, and access control.

4 REFERENCES

[1] D. E. Bell and L. LaPadula, "Secure Computer Systems: Mathematical Foundations and Model, M74-244, MITRE Corp. Bedford, MA, 1973

[2] Mike Burmester, Breno DeMederios, and Alec Yasinsac, "Community-centric vanilla-rollback access…", 13th International Workshop on Security Protocols, April 20-22, 2005

[3] D. Denning, "A Lattice Model of Secure Information Flow," *Communications of the ACM* 19 (5), pp. 236-243 (May 1976)

[4] M. A. Harrison, W. L. Ruzzo and J. D. Ullman. Protection in Operating Systems, *Communications of ACM*, Volume 19. No. 8. August 1976.

[5] Anita Jones, "Protection Mechanism Models: Their Usefulness", In *Foundations of Secure Computation*, 1978, pp. 237-252

[6] Ravi S. Sandhu, Edward J. Coynek, Hal L. Feinsteink and Charles E. Youmank, "Role-Based Access Control Models", *IEEE Computer*, Volume 29, Number 2, February 1996, pp. 38-47

[7] *International Workshop on Ubiquitous Access Control*, July 17-21, 2006 - San Jose, California, USA, http://www.mobiquitous.org/

[8] The Second International Workshop on Security in Ubiquitous Computing Systems (SecUbiq-06), August 1-4, 2006, Seoul, Korea

[9] Dieter Gollman, "What do we mean by Entity Authentication?", In Proceedings of the *IEEE 1996 Symposium on Research in Security and Privacy*, pages 46--54. IEEE, 1996

[10] R. R. Jueneman, S. M. Matyas, and C.H. Meyer, "Message Authentication", *IEEE Communications Magazing*, Vol. 23, No. 9, September 1985

[11] Martin Abadi, Cedric Fournet, Georges Gonthier, "Authentication Primitives and their Compilation", *Proc. of the 27th ACM Symp. on Prin. of Prog. Lang.* (Jan. 2000), 302-315.

[12] Armin Liebl, "Authentication in Distributed Systems: A Bibliography", Operating Systems Review, 27(4):31--41, October 1993

[13] Roger M. Needham, Michael D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", Comm. of the ACM, Dec, 1978, Vol. 21, N0. 12, pp. 993-999

[14] Burrows, M., Abadi, M., and Needham, R. M. "A Logic of Authentication", In *Proceedings of the Royal Society of London*, A 426:233-271, 1989

[15] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in Distributed Systems: Theory and Practice", ASM OS Review, Vol 25, No. 5, pp. 165-182

[16] W. Diffie, P. C. van Oorshot, and M. J. Wiener, "Authentication and Authenticated Key Exchanges", *Designs, Codes and Cryptography*, 2(2):107-125, June 1992

[17] Ray Bird, Inder Gopal, et al. "Systematic Design of a Family of Attack Resistant Authentication Protocols", *IEEE Journal on Selected Areas in Comm.*, Vol. 11, No. 5, June 1993

[18] Wm. A. Wulf, Alec Yasinsac, Katie S. Oliver, and Ramesh Peri, "Remote Authentication Without Prior Shared Knowledge", *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, February 2-4, 1994, San Diego, Ca., pp. 159-164

[19] Gavin Lowe, "Casper: A Compiler for the Analysis of Security Protocols", Journal of Computer Security, Volume 6, pp 53-84, 1998.

[20] Y. Desmedt and Y. Frankel, "Threshold Cryptosystems," In Crypto 89, Springer-Verlag Lecture Notes in Computer Science (Vol. 435), pp. 307-15, 1990

[21] Independent Review of the Carnivore System, Final Report, Contract No. 00-C-0328, IITRI CR-030-216, IIT Research Institute, 8 December, 2000

[22] Dorothy E. Denning, "An Intrusion-Detection Model," Proceedings of the 1986 *IEEE Symposium on Security and Privacy*, p. 118

[23] K. Biba, "Integrity Considerations for Secure Computer Systems," Technical Report MTR-3153, MITRE Corporation, Bedford, MA (Apr. 1977)

[24] D. Clark and D. Wilson, "A Comparison of Commercial and Military security Policies, "*Proceedings of the 1987 Symposium on Security and Privacy,* pp. 184-194, (Apr. 1987)

[25] D. Brewer and M. Nash, "The Chinese Wall Security Policy," *Proceedings of the 1989 Symposium on Security and Privacy,* pp. 206-214 (May 1989)

[26] J. Douceur. "The Sybil Attack," In *Proceedings of the 1stInternational Workshop on Peer-to-Peer Systems,* (IPTPS), 2002

[27] J. Marshall, V. Thakur, and A. Yasinsac, "Identifying Flaws in the Secure Routing Protocol", Proc. of 22nd Intl. Perf., Comp., and Comm. Conf., Apr. 9-11, 2003, pp. 167-174

Semantic Digital Rights Management for Controlled P2P RDF Metadata Diffusion

Roberto García, Giovanni Tummarello

GRIHO – Human-Computer Interaction Research Group
Universitat de Lleida, Spain - roberto@griho.net
SEMEDIA – Semantic Web and Multimedia Group
http://semedia.deit.univpm.it - g.tummarello@gmail.com

Since the early works in the W3C Semantic Web initiative, RDF has been generically indicated as a potential basis for legally binding exchange of semantically structured information. In this paper we introduce and detail a procedural framework that could support such legally binding exchange. The proposed methodology is based on a Copyright Ontology, a copyright conceptualisation which includes concrete rights expression languages like MPEG-21 REL, and RDF model decomposition based on the Minimum Self Contained Graph theory. The procedure seems particularly useful when applied to P2P semantic web scenarios.

1. Introduction

The knowledge representation capabilities of RDF are agnostic with respect to the content and the purpose for which it is used. Since the early works in the W3C Semantic Web initiative, however, a few use cases stood out and among these there was the idea that RDF might have been potential basis for legally binding exchange of semantically structured information [1].

In this paper we address a scenario which is becoming more and more common on both the Semantic Web and in “Web 2.0” websites; information does not simply go directly from the source to the intended destination. Instead, information is *mashed up*, *aggregated*, *filtered*, *republished*, *annotated*, etc. This happens notably with RSS feeds but more on the “Semantic Web”, with frameworks such as DBin [2] where peers collect bits of RDF (related to resources of common interest) which can then be redistributed either to other peers or web republished.

Clearly however, not all data sources would in any case agree on uncontrolled use and redistribution of their produced content. For example, a stock price web service might be willing to provide real time information to a subscriber as long as “it is not publicly redistributed before 10 minutes”. Similarly, in a DBin P2P RDF group, a user might want to give information to other peers “as long as it is redistributed only to those who have a verified @deit.univpm.it address”.

In such scenarios, simple access control to the information sources (e.g. password protected) does not suffice and a non machine readable licence (e.g. a fixed licence that one has to agree with a “I understand the terms and condition” checkbox at sign up time) would not allow any automatic and dynamic handling of such information distribution scenarios.

The procedure we discuss in this paper addresses such needs and enables a source peer (from here on *source*) to provide a piece of RDF to a receiving peer (*receiver*) in a manner which could provide the technical basis for legal protection.

2. The proposed exchange procedure: outline

In this section we describe the procedure by which the source provides RDF to the receiver along with a licence which specifies how such information may be used. The procedure involves multiple steps requires trust of the identity of the remote party, i.e. the parties must know or have a way to track the legal identity of the creator of the public key that will verify the signing of the licences. There are many ways by which this can be achieved (e.g. via a certifying third party like for example V*erisign*) so the discussion of these is outside the scope of this paper.

For the rest of the discussion we will use the term *cite* to indicate a pointer to the information, e.g. an URL. A *non dereferenciable citation* is a citation by the way of, for instance, a digital hash: a receiver can check that it refers to the information just when it has the information itself or via a third party. With the term *quote* we indicate providing the information itself along with additional control information.
In time steps, the exchange proceeds as follows:

1) **R makes a request to S**. As a result of such request R expects S to give information expressed in RDF. *Optionally*: The request is digitally signed so to provide R with a way to make a "personalized" licence offer
2) **S receives the request**, creates the RDF for the answer and uses the minimum self contained graph (MSG) decomposition as highlighted in the next chapter to obtain a set of digital hashes which enable to *cite* in a non dereferenciable way the information it is willing to give. Uses the hashes in a licence created with the methodology described in section 3 and sends the result, from here on called *proposal,* to R. *Optionally*: signs the proposal so to provide S with the guarantee that if agreed, the answer will actually be provided within the specified terms
3) **R receives the proposal** and, if it decides that the terms are agreeable, signs it and returns it to S. *Optionally*: thanks to the properties of MSGs, R can check if the answer correspond to information which is already locally known. In this case R could drop the request as not interesting, or proceed, e.g., in case it is important for R to prove that the information was in fact legally acquired.
4) **S receives the signed proposal,** stores it and replies with the answer computed in 2). *Optionally*: the signed proposal might be countersigned to allow R to prove that the information was obtained by legal means.

2.1. An introduction to the Minimal Self Contained Graph theory

In this section we will illustrate the Minimum Self Contained Graph (MSG) theory. The discussion will deepen that first illustrated in [3] and will provide the bases for the understanding precisely the procedure.

Let's first define what is the minimum “standalone” fragment of an RDF model. As blank nodes are not addressable from outside a graph, they must always be considered together with all surrounding statements, i.e. stored and transferred together with these. MSG are the smallest components of a lossless decomposition of a graph which does not take into account inference such as provided by OWL, as concepts such as RDF-Molecules show [4] We will here give a formal definition of MSG (minimum Self-contained Graph) and will cite some important properties (for proofs, see [3]).

Def 1. An RDF statement involves a name if it has that name as subject or object.

Def 2. An RDF graph involves a name, if any of its statements involves that name.

Def 3. Given an RDF statement s, the Minimum Self-contained Graph (MSG) containing that statement, written MSG(s), is the set of RDF statements comprised of the statement in question and, recursively, for all the blank nodes involved by statements included in the description so far, the MSG of all the statements involving such blank nodes;

It is possible to show however that the choice of the starting statement is arbitrary and this leads to a unique decomposition of the RDF graph into MSGs.

It is also possible to prove that:

Theorem 1. If s and t are distinct statements and t belong to MSG(s), then MSG(t) = MSG(s).

Theorem 2. Each statement belongs to one and only one MSG.

Corollary 1. An RDF model has a unique decomposition in MSGs.

This is a consequence of theorem 2 and of the determinism of the procedure.

As a consequence of the Corollary 1, a graph can be incrementally transferred between parties by decomposition into MSGs and transfers with granularity down to one MSG at a time. Such transfer would be, as consequence of theorem 2, maximally network efficient as statements would never be repeated.

Definition 4. The RDF Neighbourhood (RDFN) of a resource is the graph composed by all the MSGs involving the resource itself.

Content based identifiers for MSGs

MSGs are standalone RDF graphs. As such they can be processed with algorithms such as canonical serialization. We use an implementation of the algorithm described in [5], which is part of the RDFContextTools Java library [6], to obtain a canonical string representing the MSG and then we hash it to an appropriate number of bits to reasonably avoid collisions. This hash acts as a unique identifier for the MSG with the fundamental property of being *content based*, which implies that two remote peers would derive the same ID for the same MSG in their DB. Sets of such IDs are used to identify the information covered in the licences.

3. Semantic Digital Rights Management

Lately, there have been great works and debate surrounding Digital Rights Management, or DRM. A DRM system (DRMS) is composed of IT components and services along with corresponding law, policies and business models which strive to enable controlled distribution of content and associated usage rights.

It is important for different DRMSs to interoperate. One of the main initiatives for DRM interoperability is the ISO/IEC MPEG-21 standardisation effort. The main interoperability facilitation components are the Rights Expression Language (REL), which is based on a XML grammar and so syntax-based, and the MPEG-21 Rights Data Dictionary (RDD) which captures the semantics of the terms employed in the REL [7]. This one, however, does so without defining a formal semantics [8].

The limitations of a purely syntactic approach and the lack of formal semantics can be overcome using a semantics based approach based on ontologies [9]. Web ontologies are used in order to benefit from the Semantic Web initiative efforts and facilitate its integration in the Web context. The Copyright Ontology [10], of which we give here an overview, is a conceptualisation effort based on OWL.

The copyright domain is a very complex one and its conceptualization is a very challenging task. In order to facilitate this, the Copyright Ontology conceptualisation task has been divided in three parts. Each part concentrates on a portion of the problem. The conceptualisation starts from building a model for the more primitive part, the Creation Model. Then, the following step is to build the Rights Model, and, finally, the Action Model on the roots of the two previous ones. This section just sketches the main points of these three models. For more details, see [11].

The Creation Model defines the different forms a creation can take. These can be classified on the three top categories common in many upper ontologies**: Abstract**, a mental concept, **Object**, a continuant or endurant and **Process**, an occurrent or perdurant. [12].

The Rights Model follows the World Intellectual Property Organisation (WIPO) recommendations in order to define the rights hierarchy. There are the economic rights plus the moral rights, as promoted by the WIPO and adopted by all the countries adhered to the Berne Convention [13].

The more relevant rights in the DRM context are the economic rights as they are related to productive and commercial aspects of copyright. The Action Model corresponds to the primitive actions that can be performed on the concepts defined in the Creation Model and which are regulated by the rights in the Right Model.

For instance, for the economic rights, these are the actions governed by them:

- **Reproduction Right**: *reproduce*, commonly speaking *copy*.
- **Distribution Right**: *distribute*. More specifically *sell*, *rent* and *lend*.
- **Public Performance Right**: *perform*; it is regulated by copyright when it is a public performance and not a private one.
- **Fixation Right**: *fix*, or *record*.
- **Communication Right**: *communicate* when the subject is an object or *retransmit* when communicating a performance or previous communication, e.g. a re-broadcast. Other related actions, which depend on the intended audience, are *broadcast* or *make available*.

- **Transformation Right**: *derive*. Some specialisations are *adapt* or *translate*.

The action concepts are complemented with a set of relations that link them to the action participants. The relations are adopted from the linguistics field and they are based on case roles [14].

The previously introduced pool of primitive actions can be combined in order to build different value chains in the copyright domain. It is complemented with a set of axioms that restrict the ways actions, rights and creation types are related.

The P2P RDF metadata diffusion scenario is governed by the Reproduction and Communication Rights. The Reproduction Right governs the *Copy* action that reproduces a piece of metadata from Peer A, where the piece resides originally, to Peer B, where the piece also resides when the copy is completed.

The Communication Right governs the generic action *Communicate*. This action corresponds, among others, to the situation where the agent responsible for a peer makes content available to others from the place and time individually chosen by them. Therefore, in the context of P2P diffusion, this is the right required by a peer in order to make a piece of metadata available for others to copy.

In order to complete the action model, there are also the licensing actions: *Agree* and *Disagree*, the building blocks for any license, as the one shown in. Fig. 1.

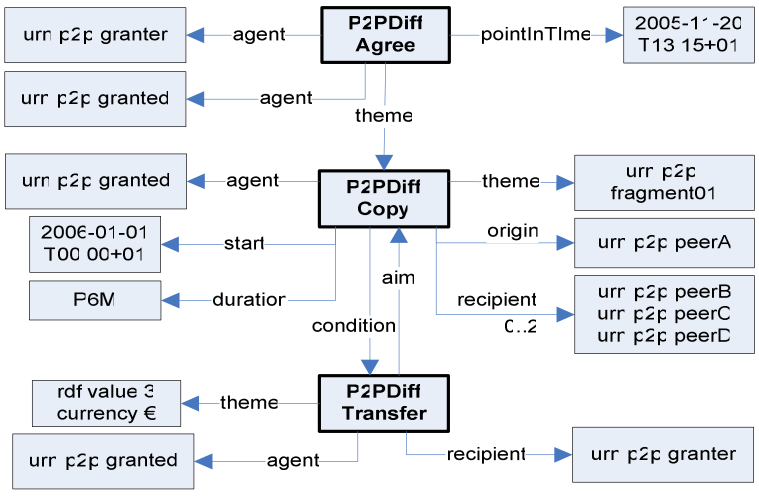


Fig. 1. Model for an agreement on a copy action pattern plus a condition

The deontic operators are implicit in the agreement model. The agreement *theme* corresponds to an implicit permission, i.e. the theme of an agreement is permitted. The *condition* on the agreement theme corresponds to an obligation, i.e. in order to fulfil the theme action it is necessary to satisfy the pattern defined by the condition property object. Finally, it is also possible to model prohibitions. This can be done in two ways, by agreeing on a negated pattern or by using the *Disagree* action.

3.1. License Checking, an example

The main objective has been to provide a straightforward and efficient implementation geared towards an extensive use of DL (Description Logic) reasoners.

Licenses are modelled as OWL Classes and copyrighted content intended uses are modelled as instances. In order to check if a usage (instance) is authorised by a set of licenses (classes) a DL reasoner is used to classify the instance in the available classes. If the instance is classified into a class that models an agreement, the *Agree* class as specified in the Copyright Ontology, the usage is authorised.

Suppose, for example, that we want to model a license that allows the agent "granted" to copy the metadata "fragment01" from "peerA" to either "peerB", "peerC" or "peerD". Additional restrictions are that at most it can be simultaneously copied to 2 peers (as a result of an individual copy action) and that the copy can be performed from January 1st 2006 to June 30th 2006.

Table 1 shows the class pattern for the *theme* values of the license *Agree*. The pattern is for *Copy* actions, so it is a subclass of *Copy*, and it is equivalent to the class resulting from the intersection of four OWL restrictions, which constitute the necessary and sufficient conditions that would trigger the classification of authorised usage instances.

Table 1. Class pattern for the actions authorised by the example license

$\text{Pattern} \sqsubseteq$	Copy	(1)
$\text{Pattern} \equiv$	$\forall \text{pointInTime}.\geq \text{2006-01-01T00:00:00}, \leq \text{2006-06-30T23:59:59} \sqcap$	(2)
	$\exists \text{agent}.\{\text{granted}\} \sqcap \exists \text{origin}.\{\text{peerA}\} \sqcap \exists \text{theme}.\{\text{fragment0001}\} \sqcap$	(3)
	$(\leq 2 \text{ recipient}) \sqcap$	(4)
	$\forall \text{recipient}.\{\text{peerC, peerD, peerB}\}$	(5)

3.2. Implementation

The Semantic DRMS is implemented at two levels. The ground level is about OWL-DL and can be implemented with a common Description Logic reasoner. Pellet[1] has been selected because it can reason over custom data types and this has been very useful to check licensing time ranges.

This however must be complemented with a metalevel that implements the deontic aspects that are implicit in the conceptual model. This metalevel guides the DL checks that have to be performed in order to capture the semantics of the implicit obligations, permissions and prohibitions. The metalevel has been also implemented programmatically.

MSG theory and tools has been implemented in [2] based on the Jena and the Sesame toolkits. The entire procedure as described in this paper is covered in the implementation of an upcoming version of the DBin platform [2] but it will be made available as a standalone library to be used embedded in other applications which exchange RDF.

[1] Pellet OWL Reasoner, http://www.mindswap.org/2003/pellet

4. Conclusions

The copyright ontology constitutes a complete framework for representing copyright value chains and the associated flow of rights situations, agreements, offers, etc. This general framework can be specialised and used in conjunction with the Minimum Self Contained RDF graph theory to implement a P2P RDF diffusion mechanism which could form a base for legally binding agreements.

The proposed methodology works based on typical semantic web tools. Licences are implemented as an OWL-DL ontology so an implementation only needs a Description Logic classifier to determine if an action is permitted.

One could say that the proposed approach would be limited to the case of protection against "verbatim" redistribution of information. While this is the case technically (MSG IDs would change with any simple modification, e.g., the insertion of a meaningless triple attached to any blank node), this does not change the validity and applicability of the procedure. It is in fact long established that copyright laws protect not only the exact representation of the protected work but also derived representations. The case is similar to one licensing a photo from a collection, changing a single pixel and wanting to redistribute it as one's own production outside fair use limits.

We believe this work can have wide applicability and cover real world requirements. The development of this idea was in fact motivated by the need to support much requested use cases in the Semantic Web P2P framework of DBin. As per DBin version 0.4, information is in fact exchanged just based on a URI based request. Under this condition, all that is known by a peer which involves that URI (at MSG level) is shipped to the requesting peer. Thanks to the procedure we propose in this paper it will be now possible to support important use cases involving information which should be exchanged but just in controlled conditions.

4.1. Related Work

While we consider DRM a natural approach for the purpose of this paper, there exist several general policy system which have been applied to SW scenarios. Ontology-based approaches rely on the expressive capabilities of Description Logic languages, such as OWL. DL reasoners can be then used to classify policies and contexts and enable deductive inferences for policy checking.

This is the approach for the Copyright Ontology implementation presented in this paper. A generic policy language also following this approach is KAoS [15] which can reason about licenses by ontological subsumption. KAoS requires however OWL-Full reasoning capabilities and its implementation is based on a theorem prover.

In contrast, rule-based approaches take the perspective of Logic Programming to encode policies as rules with variables. Rei is a policy framework based on rules [16]. Rules are expressed as triples following a pattern that is typical of logical languages like Prolog. In fact, Rei is developed using the XSB Prolog engine. Rei overcomes the variables limitation and enables the definition of policies that refer to dynamically determined values. However, this prevents it from exploiting the full potential of the

OWL language. In fact, Rei rules knowledge is treated separately from OWL ontology knowledge due to its different syntactical form.

To overcome the limitations of this trade-off between ontology and rule-based policies, some have proposed a hybrid solutions [17]. This is also the choice for the Copyright Ontology implementation, as in fact SWRL is used for some axioms and for metalevel reasoning.

References

1. Resource Description Framework (RDF): Concepts and Abstract Data Model. W3C Working Draft 2002. RDFhttp://www.w3.org/TR/2002/WD-rdf-concepts-20020829
2. G. Tummarello, C. Morbidoni, P. Puliti, F. Piazza, "The DBin Semantic Web platform: an overview", WWW2005 Workshop on The Semantic Computing Initiative (SeC 2005)
3. Tummarello G.,;Morbidoni C.; Puliti P; Piazza F. "Signing individual fragments of an RDF graph" , 2005, World Wide Web Conference 2005 Poster Track
4. Ding L.; Finin, T; Peng, Y; Pinheiro da Silva, P; , McGuinness, D , "Tracking RDF Graph Provenance using RDF Molecules" , 2005, Proceedings of the Fourth International Semantic Web Conference, November 2005
5. Carroll, J "Signing RDF Graphs", 2003, International Semantic Web Conference 2003
6. Tummarello, G.; Morbidoni C.; "RDFContext Tools 0.2", http://semedia.deit.univpm.it/tiki-index.php?page=RdfContextTools
7. Wang, X.; DeMartini, T.; Wragg, B.; Paramasivam, M.; Barlas, C.: "The MPEG-21 rights expression language and rights data dictionary". IEEE Transactions on Multimedia, Vol. 7, No. 3, pp. 408-417, 2005
8. García, R.; Delgado, J.: "An Ontological Approach for the Management of Rights Data Dictionaries". In Moens, M. & Spyns, P. (ed.): "Legal Knowledge and Information Systems". IOS Press, Frontiers in Artificial Intelligence and Applications Vol. 134, 2005
9. García, R.; Gil, R.; Delgado, J.: "A Web Ontologies Framework for Digital Rights Management". In press, Journal of Artificial Intelligence and Law, Springer, 2006
10. Copyright Ontology, http://rhizomik.net/ontologies/copyrightonto
11. García, R.: "A Semantic Web Approach to Digital Rights Management". PhD Thesis, Technologies Department, Universitat Pompeu Fabra, Barcelona, ES, 2006. http://rhizomik.net/~roberto/thesis
12. Niles, I.; Pease, A.: "Towards a Standard Upper Ontology". In Welty, C.; Smith, B. (eds.): Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS), Maine, USA, 2001
13. Berne Convention, http://www.wipo.int/treaties/en/ip/berne
14. Sowa, J.F.: "Knowledge Representation. Logical, philosophical and computational foundations". Brooks Cole Publishing Co., 2000
15. Uszok, A., et al.: "KAoS policy management for semantic web services". IEEE Intelligent Systems, Vol. 19, Num. 4, pp. 32-41, 2004
16. Kagal, L.: "A Policy Based Approach to Governing Autonomous Behavior in Distributed Environments". PhD Thesis, University of Maryland, Baltimore County, USA, 2004
17. Bradshaw, J.; Kagal, L.; Montanari, R.; Toninelli, A.: "Rulebased and ontology-based policies: Toward a hybrid approach to control agents in pervasive environments". In Proceedings of the ISWC2005 Semantic Web and Policy Workshop, 2005

Author Index