MOPAS 2011

The Second International Conference on Models and Ontology-based Design of Protocols, Architectures and Services

April 17-22, 2011

Budapest, Hungary

MOPAS 2011 Editors

Michel Diaz, LAAS, France

Ernesto Exposito, LAAS, France

MOPAS 2011

Foreword

The Second International Conference on Models and Ontology-based Design of Protocols, Architectures and Services [MOPAS 2011], held between April 17 and 22 in Budapest, Hungary, proposed a new context for presenting achievements, surveys and perspectives in the areas of design, architecture and implementation based on ontologies and related models.

We take here the opportunity to warmly thank all the members of the MOPAS 2011 Technical Program Committee, as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to MOPAS 2011. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the MOPAS 2011 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that MOPAS 2011 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the areas of models and ontology-based design of protocols, architectures and services.

We are convinced that the participants found the event useful and communications very open. We also hope the attendees enjoyed the historic charm of Budapest, Hungary.

MOPAS 2011 Chairs:

Michel Diaz, LAAS, France
Ernesto Exposito, LAAS, France
Raj Jain, Washington University in St. Louis, USA
Arun Prakash, Fraunhofer Institute for Open Communication Systems (FOKUS) - Berlin, Germany

MOPAS 2011

Committee

MOPAS Chairs

Michel Diaz, LAAS, France
Ernesto Exposito, LAAS, France
Raj Jain, Washington University in St. Louis, USA

MOPAS Industry/Research Chair

Arun Prakash, Fraunhofer Institute for Open Communication Systems (FOKUS) - Berlin, Germany

MOPAS 2011 Technical Program Committee

Mourad Alia, Orange Business Services - Grenoble, France
Agnieszka Brachman, Silesian University of Technology, Poland
Hakki Candan Cankaya, University of Texas at Dallas, USA
Olivier Curé, Université Paris-Est, France
Michel Diaz, LAAS, France
Petre Dini, IARIA, USA / Concordia University, Canada
Rachida Dssouli, Concordia University, Canada
Ernesto Exposito, LAAS, France
Bin Guo, IT-SudParis, France
Robert C. H. Hsu, Chung Hua University, Taiwan
Zahid Iqbal, University Graduate Center (UNIK) - Kjeller, Norway
Raj Jain, Washington University in St. Louis, USA
Brigitte Jaumard, Concordia University - Montreal, Canada
Achilles Kameas, Hellenic Open University-Patras, Greece
Marc Lacoste, Orange Labs, France
Thomas D. Lagkas, University of Western Macedonia, Greece
Myriam Lamolle, IUT de Montreuil, France
Vlad Nicolicin Georgescu, Université de Nantes / SP2 Solutions - La Roche sur Yon, France
Peera Pacharintanakul, TOT, Thailand
Arun Prakash, Fraunhofer Institute for Open Communication Systems (FOKUS) - Berlin, Germany
Neeli R. Prasad, Aalborg University, Denmark
Samir Sebbah, Concordia University, Montreal, Qc, Canada
Shensheng Tang, Missouri Western State University - St. Joseph, USA
Saïd Tazi, LAAS-CNRS. Université Toulouse 1, France
Zoltan Theisz, Ericsson Ireland Ltd., Ireland
Dimitrios D. Vergados, University of Piraeus, Greece
Krzysztof Walkowiak, Wroclaw University of Technology, Poland
Roberto Willrich, Santa Catarina Federal University, Brazil
Hirozumi Yamaguchi, Osaka University, Japan
Nataša Živic, University of Siegen, Germany

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

The Design, Instantiation, and Usage of Information Security Measuring Ontology

Antti Evesti, Reijo Savola, Eila Ovaska, Jarkko Kuusijärvi
VTT Technical Research Centre of Finland
Oulu, Finland
e-mail: antti.evesti@vtt.fi, reijo.savola@vtt.fi, eila.ovaska@vtt.fi, jarkko.kuusijarvi@vtt.fi

***Abstract*—Measuring security is a complex task and requires a great deal of knowledge. Managing this knowledge and presenting it in a universal way is challenging. This paper describes the Information Security Measuring Ontology (ISMO) for measuring information security. The ontology combines existing measuring and security ontologies and instantiates it through example measures. The ontology provides a solid way to present security measures for software designers and adaptable applications. The software designer can utilise the ontology to provide an application with security measuring capability. Moreover, the adaptable application searches for measures from the ontology, in order to measure a security level in the current run-time situation. The case example illustrates the design and run-time usage of the ontology. The experiment proved that the ontology facilitates the software designer's work, when implementing security measures for applications that are able to retrieve measures from the ontology at run-time.**

Keywords-adaptation; run-time; quality; measure; security metric; software

I. INTRODUCTION

Software applications running on devices and systems may face needs for changes due to alterations happening in their execution environments or intended usages. These changes may have a considerable effect on the security requirements of the software system. Moreover, emerging security threats and vulnerabilities may affect the achieved security level. However, the software system is required to achieve a desired security level in these changing circumstances [1]. Therefore, the software has to be able to observe the security level at run-time, measure the fulfilment of the security requirements, and adapt itself accordingly. However, measuring the security level at run-time requires the correct measures and measurement techniques for each situation. Defining the measures and the measuring techniques is a time consuming task and requires the use of experts from different domains. Thus, it is important to present the defined measures in a universal and reusable form. In addition, problems concerning how to present these measures, the measuring techniques, and their mutual relationships have to be solved in a way that facilitates run-time security measuring. Ontologies provide a possibility to manage this knowledge, making it possible to describe different security requirements and ways to measure the fulfilment of these requirements.

Ontologies are utilised in [2] to achieve the required quality of the software at a design-time. Thus, it is reasonable to utilise ontologies as a knowledge base for quality management at the run-time. Furthermore, the work in [3] presents the architecture for developing software applications with security adaptation capabilities – the presented approach assumes that the knowledge required for security monitoring and adaptation is available from ontologies.

In this work, we will present a novel ontology for the run-time security measuring – called Information Security Measuring Ontology (ISMO). ISMO combines a terminology from a software measurement area in general and the security related terminology. In addition, a few security measurements are added to the ontology in order to instantiate it. The novelty of our work comes from this combination – based on our current knowledge, there isn't any other ontology which describe security measuring in a run-time applicable way. The content of ISMO can be enhanced after the software application has been delivered. Hence, the measuring process is based on the up-to-date specifications of security measures. The purpose of this new ontology is to make it possible for software applications to utilise security measures during run-time in changing environments. Therefore, it is possible for the application to measure the fulfilment of its security requirements and adapt the used security mechanisms if the required security is not met. In other words, the measuring acts as a trigger for the adaptation. However, to achieve software applications with a measuring capability, the developer must have implemented a set of measuring techniques as a part of it. Thus, ISMO also provides input for developers – presenting what measuring techniques are to be implemented and how.

The remainder of the paper is organised as follows: Section 2 provides background information; Section 3 presents an overview of the combined ontology and mentions some security measurements. Section 4 instantiates the defined ontology. Section 5 explains how the ontology is utilised. A case example is presented in Section 6. Finally, a discussion and conclusions close the paper.

 ISBN: 978-1-61208-130-4

II. BACKGROUND

ISO/IEC defines security in [4] as follows: "The capability of the software product to protect information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them." Furthermore, in some sources security is thought to be a composition of confidentiality, integrity and availability [5, 6]. In [7], these security sub-attributes are called security goals.

Zhou [8] defines ontology as a shared knowledge standard or knowledge model, defining primitive concepts, relations, rules and their instances, which comprise topic knowledge. It can be used for capturing, structuring and enlarging explicit and tacit topic knowledge across people, organizations, and computer and software systems.

Blanco et al. [9] lists several security ontologies in their work. In addition, our earlier work [10] also compares a number of security ontologies, particularly those that are applicable for run-time usage. It is noticed in [10] that security ontologies for run-time usage exist – especially for service discovery and matchmaking, for example, ontologies from Denker et al. [11] and Kim et al. [12]. In addition, security ontologies which concentrate on software design and implementation phases also exist, e.g., works from Savolainen et al. [13] and Tsoumas et al. [14]. From these ontologies, only the work from Savolainen et al. [13] takes measurements into account, by presenting a high level classification for different security measures. However, the most extensive information security ontology at the moment is the one proposed by Herzog et al. [7], called an ontology of information security – abbreviated as (OIS) in this work. The OIS is intended to provide a general vocabulary or an extensible dictionary of information security. It is applicable at design and run-time alike, and it contains more concepts than all the above mentioned security ontologies altogether. Thus, this ontology provides a sound starting point for defining the concepts of ISMO.

The OIS does not contain concepts for describing measures. Therefore, the Software Measurement Ontology (SMO) [15] is utilised for measurement definitions. The SMO presents the generic measurement terminology related to software measurements. In other words, ontology is quality attribute independent. The SMO collects and aligns terminology from several standards of software engineering, software quality metrics, and general metrology. It is important to notice that the SMO uses a term measure instead of metric. Thus, the measure term will be used in this work. The SMO divides measures into three sub-classes: namely a base measure, derived measure, and indicator – all of which inherit the same relationships from other concepts. The base measure is an independent 'raw' measure. A derived measure is a combination of other derived measures and / or base measures. Finally, an indicator can be a combination of all of these three types of measures. The complexity of these measures increases when moving from base measures to derived measures and further on to indicators. In literature, base measures and derived measures are also called direct and indirect measures; however we will follow the terminology defined by the SMO.

Hence, this work draws mappings between the OIS and SMO, instead of defining a new ontology from scratch. This is considered to be reasonable since a remarkable effort has been invested into these existing ontologies and both are scientifically reviewed and accepted. In addition, the reuse of existing ontologies is suggested in [16] as one potential approach for ontology development.

III. THE DESIGN OF INFORMATION SECURITY MEASURING ONTOLOGY

This section describes how the combined ontology ISMO is designed. SMO [15] contains 20 generic measurement related concepts and their relationships. Thus, security measures will be used to instantiate ontology for security measuring purposes – creating base measures, derived measures, and indicators. On the other hand, OIS [7] contains concepts related to threats, assets, countermeasures, security goals, and the relationships between those concepts. In addition, the OIS describes a couple of vulnerabilities and how these act as enablers for threats. The OIS already contains some of these concepts as an instantiated form, such as the security goals of authentication, integrity, etc.

The purpose of combining these two ontologies is to achieve an ontology that makes it possible to measure the fulfilment of security requirements, i.e., security goals and levels. In other words, the purpose is to enable an operational security correctness measurement, as called in [17]. Therefore, the requirements are described by a means of vocabulary from the OIS. The requirements fulfilment is measured with indicators – which combine several measures – defined in the SMO. The security measures, i.e., indicators, are different for each security goal, e.g., a level of authentication or non-repudiation is measured with different measures. However, these measures can utilise the same base measures. On the other hand, the same security goal can be achieved with different countermeasures, which in turn might require their own measures. For instance, the authentication level, which is achieved, is measured in a different way when a security token is used instead of a password authentication. Hence, there are only a few concept-to-concept mappings between these two ontologies, but additional mappings appear when the measures are instantiated. By using a terminology from ontologies, a mapping refers to the property between the concepts. Adding mappings for instantiated measures requires domain expertise, i.e., the capability to recognise applicable measuring techniques for a particular security goal and related mechanisms. Furthermore, the mapping requires a capability to recognise threats which affect to the particular security goal and/or mechanism. Mappings at the instantiation phase are described more detail in Section IV.

Fig. 1 shows an overview of the combining process. Firstly, the mappings between the concepts are drawn. Secondly, security measure instances are added and related mappings for each measure are defined. Thus, the SMO is used as a guideline as how to define the instances of security measures.

 ISBN: 978-1-61208-130-4

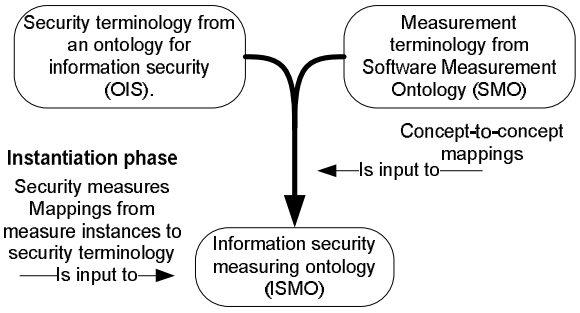

Figure 1. An overview of the combining process.

TABLE I shows mapping properties between concepts from the SMO and OIS. These are mappings made from concept to concept. Each *SecurityGoal* has an *Indicator* – intended for measuring the fulfilment of the goal. The SMO uses the term *QualityModel* for defining measurable concepts. *QualityModel* is a quality attribute dependent, i.e., security in this case. Thus, the *QualityModel* concept is related to *SecurityGoal*. The *MeasurableConcept* from the SMO is also mapped to the *SecurityGoal*, through the means of the *isDefinedFor* property.

In the SMO, *MeasurableConcept* relates to *Attribute*, meaning a characteristic that will be measured. Thus, *Attribute* can relate to countermeasures, threats, or assets, depending on the measure which is used. *hasMeasurableAttribute* is optional, meaning that mappings to the attributes are made during the phase when the measures are instantiated.

TABLE I. MAPPING PROPERTIES BETWEEN SMO AND OIS

Concept from OIS	Mapping property (direction)	Concept from SMO
SecurityGoal	hasIndicator (->)	Indicator
SecurityGoal	isRelatedTo (<-)	QualityModel
SecurityGoal	isDefinedFor (<-)	MeasurableConcept
Countermeasure, Threat, Asset	hasMeasurableAttribute (->) (optional)	Attribute

A. Security Measures

The overall security level of the product can be represented by a combination of relevant security attribute measures. However, it is not possible to cover all security measures in this work. Consequently, we will concentrate on user authentication, and thus, the ISMO is instantiated by these measures.

In [18] measures for various security goals (e.g., authentication, integrity, etc.) are defined by using a decomposition approach introduced by Wang et al. in [19]. Authentication can be decomposed into five components – called BMCs (Basic Measurable Components) – as follows: Authentication Identity Uniqueness (AIU), Authentication Identity Structure (AIS), Authentication Identity Integrity (AII), Authentication Mechanism Reliability (AMR), and Authentication Mechanism Integrity (AMI). Savola and Abie [18] define equations for these BMCs, and in addition, the equation for combining Authentication Strength (AS) from these BMCs. AS is an aggregated user-dependent measure that can be utilized in authentication and authorization.

The user-dependent AS results can be combined into a system-level AS, which can be utilized in run-time adaptive security decision-making [18]. When considering a software application that measures its security level at run-time, AIS, AII, and AMR are particularly applicable. In other words, an application cannot measure AIU and AMI, as the information which is required for these measures is only available at the server side where the application will be authenticated. Thus, in this work, the measures for the AIS will be used as example measures.

To measure AIS, we utilise a measure intended for situations where the authentication is based on a password – called the structure of the password. It is commonly understood that the structure of a password, i.e., the length and variation of the symbols, affects the achievable authentication strength. Therefore, we divide passwords into groups such as: i) a PIN code containing four numbers, ii) a password containing 5-9 lower case characters, and iii) a password containing over 10 ASCII symbols. Intuitively, *group i* provides the worst authentication level, *group ii* offers an increased authentication level, and *group iii* is the best alternative.

Another measure that we utilise for password based authentication is the age of the password, i.e., how long the same password has been used. This measure can be used for two different purposes. Firstly, to measure the security policy fulfilment, e.g., a policy can define that the password has to be changed every three months. Secondly, the measure can be utilised as a factor of measuring the authentication strength by utilising more complex analysis models. The age of the password is also mapped to the AIS from BMCs.

These two measures are simple to understand, and thus, provide a good starting point for instantiating ISMO. The graphs in Fig. 2 illustrate how the password strength is affected by the structure and age of the password. These are however merely examples and we do not claim that these affects are linear.

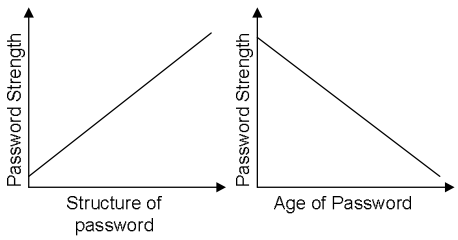

Figure 2. Conceptual correlation graphs for the authentication measures.

IV. THE INSTANTIATION OF INFORMATION SECURITY MEASURING ONTOLOGY

In this section, the authentication related measures are instantiated as a part of the ISMO and the required mappings are added. Fig. 3, Fig. 4, and Fig. 5 present the instantiated ontology – rectangles depict the concepts from SMO and ellipses refer to concepts from OIS. The name of each

 ISBN: 978-1-61208-130-4

concept is presented in the figures and separated from the instance name by a colon, i.e., ConceptName : InstanceName. The property mappings between these two ontologies are presented in bold fonts. For reasons of clarity, the instantiated ontology is presented in three separated figures. Consequently, some concepts may appear in each figure, but from a different viewpoint, i.e., presenting different properties.

The SMO contains the concept *MeasurableConcept*, which corresponds conceptually to the BMCs, which are defined in [18]. Thus, AIS BMC is instantiated in the ontology as a *MeasurableConcept*. The concept *QualityModel* refers to security goals in this work. Hence, there is a mapping property from the *QualityModel* concept to Security goals (authentication, confidentiality, etc.), as mentioned in Section 3.

The *PasswordAge* measure (Fig. 3) is the first measure instance which is added to the ISMO. In the SMO, measures are defined for attributes and these attributes are related to the measurable concept. AIS is an instance of the measurable concept and *PasswordAge* is one of the related attributes. This attribute is measured through the means of an instantiated derived measure, called *UsageTime*. Hence, the derived measure is not purely security related, and can also be applied for other attributes, e.g., the usage time of the CPU in performance measurements. The derived measure *UsageTime* is calculated with a measurement function – defining that *UsageTime* is the current date minus the starting date. The calculation of the value for this measurement function requires that a base measure instance called *Date* is used. *Date* is a base measure, meaning that it is not dependant on other measures and its value is measured by the measurement method. The measurement method for the *Date* measure is simple: taking the date value from the system clock. Defining *UsageTime* as a derived measure may seem like an overestimation. However, detailed definitions are required in order to achieve a measuring ontology that supports run-time security measuring.

The second measure – presented in Fig. 4 – is connected to the AIS via an attribute called *PasswordStructure*. The attribute is measured with an instantiated indicator called *PasswordType*. Indicators are calculated using an analysis model. In this context, the analysis model is a set of rules, which can be thought as if-then-else statements. We have decided to use statements which are very close to the natural English language, so that the analysis models could be updated without an extensive knowledge on programming. The statements of the analysis model can be updated later on to, e.g., the standard SPARQL [20] queries. The analysis model itself is saved as a string literal, so it can be easily changed into a SPARQL statement. For simplicity, the following analysis model is defined for the *PasswordType*:

- |Length < 5 AND OnlyNumbers := PINCode|
- |Length >= 5 AND Length <= 9 AND OnlyAlphabets := NormalPassword|
- |Length > 9 AND Length < 12 AND NumberOfDifferentSymbols >= 3 := GoodPassword|
- |ELSE := WeakPassword|

Thus, four base measures are used in this analysis model, i.e., *OnlyNumbers*, *OnlyAlphabets*, *Length* and *NumberOfDifferenSymbols*. These are measured using appropriate measurement methods, respectively (methods for *OnlyNumbers* and *OnlyAlphabets* are omitted from Fig. 4. In addition, these base measures are also connected to the AIS via appropriate attributes. For reasons of clarity, these are not presented in Fig. 4; however, TABLE II also lists these attributes.

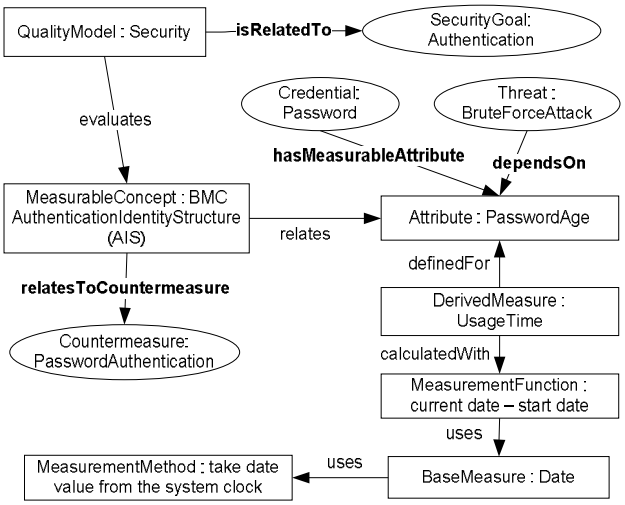

Figure 3. The age of the password.

The above mentioned attributes *PasswordStructure* and *PasswordAge* are mapped to the *BruteForceAttack* threat from the OIS. Thus, these are possible extension points in the future, in so far as risk related measures are added to ISMO. The risk measures are applied for run-time usage in [21].

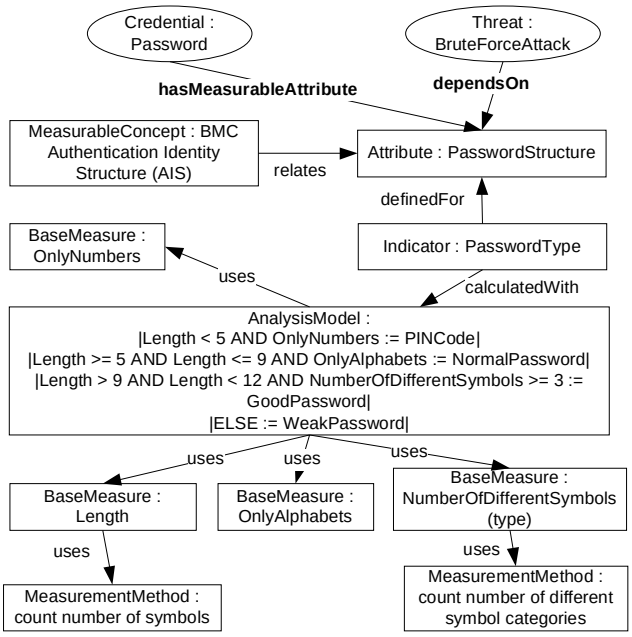

Figure 4. The structure of the password.

 ISBN: 978-1-61208-130-4

The final measure instantiated into the ontology is *AuthenticationLevel* (Fig. 5), which is an instance of the indicator concept – intended to combine the above described measures. The *AuthenticationLevel* is calculated with an analysis model in a similar manner as described for the *PasswordType* above. The analysis model is as follows:

- |PasswordType == PINCode := Level1|
- |PasswordType == NormalPassword AND UsageTime >= 180 AND UsageTime < 365 := Level2|
- |(PasswordType == GoodPassword AND UsageTime < 180) OR (PasswordType == NormalPassword AND UsageTime < 90 := Level3)|
- |ELSE := Level1|

The analysis model for the *AuthenticationLevel* uses the results from the *PasswordType* indicator and the *UsageTime* derived measure. Therefore, the calculation of the authentication level, according to this analysis mode, requires the five base measures, i.e., *OnlyNumbers*, *OnlyAlphabets*, *Date*, *Length*, and *NumberOfDifferent-Symbols*, presented in Fig. 3 and Fig. 4.

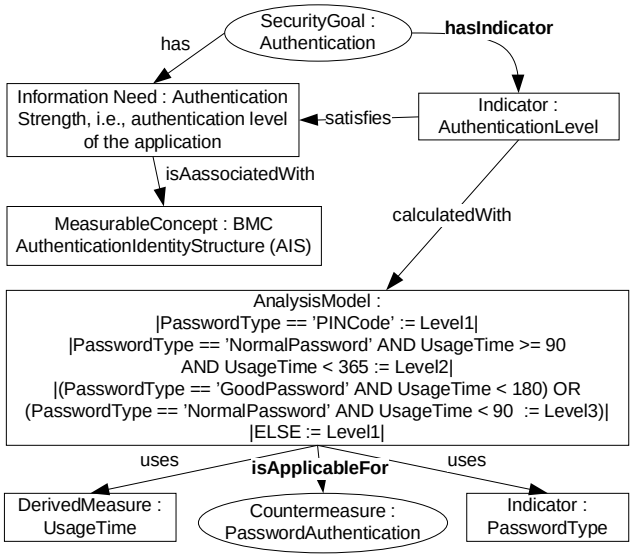

Figure 5. Authentication level.

Currently, the presented analysis models are very simple, utilising only a few base measures, and need to be enhanced in the future. Nevertheless, these analysis models provide the possibility to test the suitability of ISMO for run-time security measurements. Furthermore, the presentation of analysis models in the ontology makes it possible to modify and update them at run-time. The following table lists the mapping properties made from/to instantiated security measures. Again, these mappings are measure dependent. Hence, the addition of a new measure instance also creates new mapping properties.

TABLE II. MAPPING THE PROPERTIES OF INSTANTIATED MEASURES

Concept from OIS [7]	**Mapping property in ISMO (direction)**	**Concept from the SMO** [15]
Threat : Brute-ForceAttack	dependsOn (->)	Attribute : PasswordStructure Attribute : PasswordAge
Credential : Password	hasMeasurable-Attribute (->)	Attribute : PasswordStructure Attribute : PasswordAge Attribute : PasswordLength Attribute : NumberOfDifferentSymbols Attribute : OnlyAlphabets Attribute : OnlyNumbers
Countermeasure : Password-Authentication	relatesToCount ermeasure (<-)	Measurable concept : AuthenticationIdentity Structure
Countermeasure : Password-Authentication	isApplicableFor (<-)	Analysis model : analysis model for authentication level

V. THE USAGE OF INFORMATION SECURITY MEASURING ONTOLOGY

This section describes how the ISMO will be used at design and run-time. In addition, ontology evolution is discussed.

A. Utilisation at Design-time

The software designers have to take several issues into account when they design an application that is intended to measure its security level and adapt itself accordingly. Firstly, the required security goals are defined – such as the user authentication. Secondly, the levels for each security goal are defined, e.g., level 1 for security goals which are not very critical and level 5 for extremely critical security goals. It is notable that ISMO does not restrict the number of security levels, for example, the analysis models in the previous section utilised three levels instead of five. Thirdly, the security mechanisms to achieve the required goals are selected, e.g., a username-password pair for authentication. The micro-architecture for run-time security adaptation is presented in [3] – working as a guideline for the software developer by showing the components which are required in an adaptation applicable software.

The OIS already contains mapping properties from goals to supporting mechanisms. However, there is no possibility to define the required levels for the goals. It should be noted that the selection and implementation of security countermeasures is highly context-dependent. The ISMO draws a mapping from the security goal to the level indicator – in our case authentication level – as presented in Fig. 5. Therefore, the software designer can retrieve the base measures from ISMO, used for calculating a particular level indicator. Based on this information, she implements the measuring methods of base measures as the part of application. For example, the authentication level indicator requires five base measures and the related measurement methods as mentioned earlier, which must all be implemented to the application. However, measurement functions and analysis models that combine base measures are not hard coded to the application. Instead, a generic

 ISBN: 978-1-61208-130-4

parser and monitor components are utilised. The parser component retrieves analysis models from ISMO and parses the rules, which depict how the base measures have to be combined. The monitor component utilises these rules and calculates the security level (the authentication level at this time) from the base measures, which is utilised in the software adaptation. The generic and implementation specific parts are presented in Fig. 6. The internal design of these components is not within the scope of this paper.

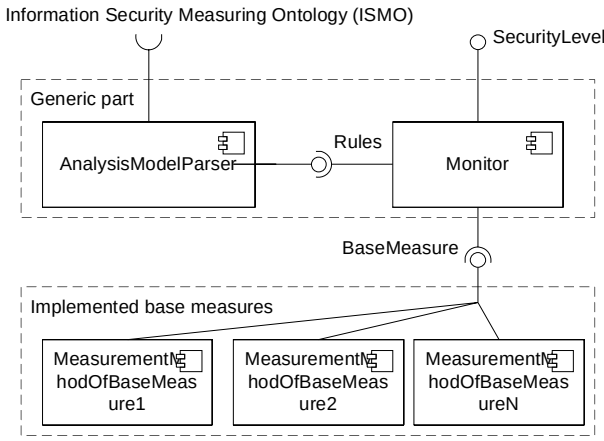

Figure 6. Generic and implementation specific parts.

B. *Utilisation at Run-time*

The application, which contains a capability to measure its security, is assumed to be aware of its security goal(s) and level(s) and how to measure the fulfilment of its goal(s). Hence, the application retrieves an indicator which is used to measure the level of a particular security goal, e.g., the authentication level indicator for the authentication goal. However, separate analysis models are required for the alternative countermeasures used to achieve user authentication. For example, Fig. 5 contains the analysis model for the password based authentication. Simultaneously, the ontology may contain an analysis model for the security token based authentication, and both of these analysis models are related to the authentication level indicator. Thus, the application must check the currently used countermeasure and select an appropriate analysis model for it from the ISMO. The *isApplicableFor* property maps the analysis model to the countermeasure and makes this selection possible.

Now, the application has a right analysis model. Based on this information, the application searches the measures that are used in the analysis model. The authentication level indicator and the related analysis model, presented in Fig. 5, use the *PasswordType* indicator and the *UsageTime* derived measure. Thus, the application queries ISMO until it finds the base measures which are required to calculate a value for the authentication level indicator. It is notable that these searches only have to be made at application start-ups and when the countermeasure is to be changed at run-time, due to security adaptation demands.

As a result of this search, the application possesses all the information which is required for measuring security. The application has the knowledge of required security goals and levels, and the base measures to be used. Therefore, the application uses measurement methods, which are implemented as a part of it during the design-time. The monitor component (in Fig. 6) combines these base measures to a security level indicator. In a situation where the application is unable to reach the required security level, it adapts the used countermeasure. The results of measuring help to recognise the part of the application that has to be adapted. The security adaptation is discussed in more detail in [21].

It is possible that the required security level changes during the application execution. For instance, the usage of the application may change in a way that requires a higher security level. This type of change does not affect the utilisation of ISMO or the measuring itself. Only the level, compared to the measurement result, changes.

C. *Ontology Evolution*

At some point, it is necessary to make changes and additions to ISMO. This is required because new threats appear, the usage of the application changes, or the environment of the application changes. The ontology evolution is a challenge from the application point of view, since ISMO is also used for making design decisions. In other words, the required base measures are selected and implemented at the design-time. Thus, a new base measure cannot appear for the application by adding it to ISMO. On the contrary, the analysis models which are used for indicators, such as the authentication level, can be dynamically changed to ISMO. For instance, the analysis model in Fig. 5 defines that level 2 is achieved with a normal password that is used for 3-12 months (90-365 days). However, this can be easily changed to the form: level 2 is achieved with a normal password that is used for 3-6 months. More complicated changes can also be made easily – the only requisite is that the application contains the required base measures. The *AnalysisModelParser* and *Monitor* components (Fig. 6) ensure that changes in the analysis models do not require any changes to the application. However, the analysis models have to be described in a common syntax that the *AnalysisModelParser* is able to parse. ISMO uses simple logical operations to combine the named measurement results, as seen in Fig. 4 and Fig. 5.

VI. A CASE EXAMPLE OF INFORMATION SECURITY MEASURING ONTOLOGY

Run-time security measuring and adaptation was earlier validated in [21, 22] – released on YouTube [23]. Now, a case example is used to exemplify both the design-time and run-time usage of ISMO. The case study takes place in a smart home environment, where the user performs different tasks with her mobile device. The RIBS platform [24] is used to build up the smart home environment. RIBS is a platform that makes it possible for heterogeneous devices to communicate with each other by a means of SIB (Semantic Information Broker) and agents. The SIB is an information

 ISBN: 978-1-61208-130-4

storage where agents publish and subscribe information. The smart home environment contains agents which publish environmental information, for instance, temperatures, humidity, etc. Home automation devices contain agents which subscribe to control information from the home SIB. Furthermore, the smart home environment contains agents, which offer entertainment information for the user, i.e., news, weather forecasts, etc.

In the case study, information from the smart home is utilised with a smart space application, which is running on a Nokia N900 mobile device. The smart space application and the related base measures are implemented using the Python programming language. In this case example, two SIBs exist. The first one (personal SIB) runs on the user's N900, as storage for ISMO. Alternatively, ISMO can be stored in the mass storage of the N900 in an OWL format. The second one (home SIB) runs on a computer in the smart home, and constitutes the home smart space. The application communicates with the personal and home SIB via TCP/IP communication and measures the achieved authentication level by a means of ISMO.

Firstly, ISMO is used at the smart space application design time as described in the previous section. The generic part, i.e., *AnalysisModelParser* and *Monitor* components, are imported to the application. The application developer makes a decision that passwords will be used for authentication and searches the supporting analysis model from ISMO. Furthermore, the base measures which are required in the analysis models are retrieved and implemented to the application. In this case, the used base measures are *OnlyNumbers*, *OnlyAlphabets*, *Length*, *NumberOf-DifferentSymbols*, and *Date*.

Secondly, ISMO is used while running a smart space application. When the user opens the smart space application, the application automatically retrieves ISMO from the personal SIB. The user then opts to join the home smart space with the smart space application. During the join operation, the user is authenticated for the first time and the authentication level monitoring process starts. The *AnalysisModelParser* component reads ISMO. The *Monitor* component receives rules on how to combine different base measures and provides the authentication level for the security adaptation. Both the *AnalysisModelParser* and *Monitor* components are running on the N900. The used countermeasure is password authentication – based on this information, the monitor component selects the correct analysis model to calculate the authentication level. It is notable that the application can contain several authentication mechanisms and ISMO provides information concerning which analysis model to use with each mechanism. The home smart space contains various types of information and the utilisation of different information requires their own authentication levels. Thus, we defined the following authentication requirements for different tasks:

- Level 1 for entertainment usage,
- Level 2 for retrieving information from sensors, etc.,
- Level 3 for controlling building automation devices.

The smart space application is aware of what the user is currently doing, i.e., it monitors the current context and reports this information to the security adaptation. The user decides to login with a username and password on authentication level 2. Thus, the user is unable to control the building automation devices. In an accelerated use case, when the password usage time reaches 12 months, the authentication level decreases to level 1. Hence, the smart space application only provides entertainment information for the user. When the user attempts to perform a task which requires higher authentication level, the smart space application recognises that an adaptation is required. The adaptation asks the user to re-authenticate with a better password, as is shown in Fig. 7. Consequently, the application user does not require any knowledge of ISMO, i.e., the smart space application seamlessly utilises the content of ISMO.

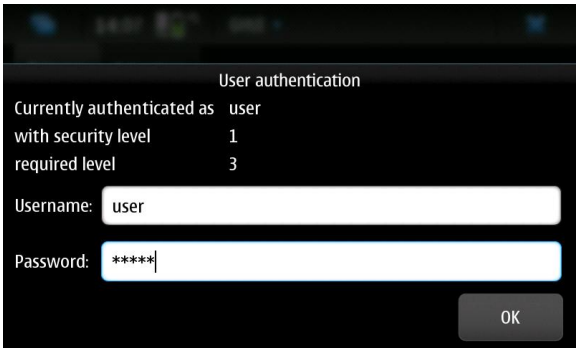

Figure 7. Re-authenticating the user.

The purpose of the case example was to test designed and instantiated ontology. Thus, the content of the analysis models and measures was not within the scope of the case. The utilisation of ISMO ensured that security can be measured in a dynamic environment. Without ISMO, the used analysis models have to be hard coded to the application, which is unreasonable in the dynamic environment. The case example proved that when the *AnalysisModelParser* and *Monitor* components exist, the implementation of the security measures to the application is straightforward. The application developer merely has to implement the required base measures as declared in the ISMO, or use existing base measures. The application developer is able to utilise measures from ISMO, without a need to implement ontology parsers. Moreover, the application was able to retrieve analysis models from ISMO and monitor the authentication level at run-time. The Monitor component calculates a new authentication level each time the used base measures change. However, the *AnalysisModelParser* component checks the content of ISMO at pre-defined intervals.

It is a commonly known issue that ontology searches may cause performance overheads. However, in this case example, the ontology was used in a mobile device without a major overhead. Nevertheless, it is important to optimise how often information is retrieved from ISMO. This helps to achieve the performance requirements of the application,

 ISBN: 978-1-61208-130-4

since changes in ISMO are only checked at pre-defined intervals. Therefore, there is no need to continually query the personal SIB. In the case example, the searches were made every 60 seconds and this kind of checking interval had no visible effects on the usability or performance of the application. Another alternative is to utilise subscriptions, which automatically inform to applications when ISMO is changed. However, the performance overhead of this option is not known beforehand, i.e., changes in ISMO can take place at anytime.

VII. DISCUSSION

In this work, we utilised existing ontologies – instead of starting from scratch – to achieve the information security measuring ontology for run-time usage. Thus, we gained a wide and extensible ontology that is compatible with its predecessors. The combination also ensures a higher maturity level, as the ontologies which were used were already validated. It can be seen from the ontology comparison presented in [10] that the existing security ontologies contain a large deal of overlapping. This work does not add overlapping concepts, which is important from a compatibility viewpoint. Utilisation of the SMO ensures that the measurement part of ISMO is generic. Therefore, the addition of new measures in the future will be easy. Furthermore, the used concepts can also be utilised to measure other quality attributes. Initially, the SMO is not intended for run-time usage. However, there are no constraints to applying the SMO at run-time situations as measuring related terminology is similar in both design and run-time measurements.

It might seem that using ontology to achieve a run-time measuring applicability is a too heavy weight solution. Nevertheless, in cases where an application contains several mechanisms for reaching a particular security goal, it will be necessary to describe the measures in detail. This is particularly necessary when the application is intended to adapt used security mechanisms. In addition, ISMO makes it possible to update and add analysis models – when a new vulnerability is found or the application usage changes. Currently, measurement functions and analysis models are described by using simple logical operations in the ontology – parsed by the *AnalysisModelParser* component. Logical operations were suitable for the measures used in this work. However, in the future, there is a need for additional mathematical operations, required in security measuring. The ontology definition is made at a level that possesses sufficient detail, and thus, it is possible for ISMO to provide the required knowledge for an autonomous measuring process.

Mapping between OIS and SMO is a complex task due to the complexity of measuring security. Currently, a concept level mapping is done, but there were only a few concept-to-concept mappings, which enforces the creation of mapping from/to instantiated security measures. Authentication related measures are instantiated to ISMO as an example. Additional mappings are required when a new measure instance is added. However, the measure instances added in this work offer an example of how to add the mappings, and thus, facilitate future additions. It is notable, that different types of measures will create entirely different mappings between these ontologies. For example, risk measures will create mappings between assets from the OIS and attributes from the SMO. On the other hand, there is not always a mapping property from the attribute concept (in SMO) to some specific credential (in OIS). Hence, mappings between these ontologies depend on the security goal, the used security mechanism, and the used measure.

The performed case example showed that ISMO can be used even in a mobile device without a major performance overhead. However, a more thorough performance evaluation has to be performed in the future. One question is how the usage of ISMO affects the achieved security. For instance, an attacker may cause constant environment changes, which in turn create a set of queries for ISMO and might jeopardize the availability. In addition, measurement methods and results might also be the target of an attack. Thus, it is necessary to perform the run-time measurement in a way that supports the achievement of security requirements, instead of creating new vulnerabilities and possibilities for attacks.

Survey of adaptive application security in [25] lists few adaptation approaches. Added to these, the Extensible Security Adaptation Framework (ESAF) [26] utilises security policies to adapt security mechanisms in the middleware layer. Furthermore, adaptation for Secure Socket Layer (SSL) is presented in [27][28] and monitoring for Java ME platform in [29]. The ISMO offers several advantages compared to existing self-adaptation and policy based approaches. Firstly, security measuring triggers an adaptation task, instead of beforehand defined situation. Secondly, ISMO is a generic solution, i.e., it is not tied to only one security mechanism, platform, or security goal. Finally, ISMO based approaches are dynamic – new analysis models can be added and the existing ones can be modified.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel ontology – called ISMO – for information security measuring, developed particularly for the needs of run-time security measuring. The main purpose was to achieve an ontology that is able to support security measuring at the run-time of an application. The ontology development utilises two existing ontologies: (i) an ontology of information security, describing security related concepts, and (ii) a software measuring ontology, describing general measuring terminology. Firstly, a conceptual mapping between these ontologies was introduced. However, security measuring is a complex task where only a few concept-to-concept relationships can be made. Secondly, the ontology was instantiated by using password related measures. The measures which were used were simple – password structure and password age – however, these measures offered a good starting point to construct ontology which is applicable to run-time security measurements. After the ontology instantiation, we described how to utilise the ISMO in a way that supports run-time measurements. The case example was utilised to exemplify how to use ISMO in a smart home environment. Finally, we

 ISBN: 978-1-61208-130-4

discussed the advantages and shortcomings related to the designed ontology.

In the future, it is important to evaluate the performance cost of using ISMO. In addition, it is important to add new security measures to ISMO, and test how easily these extensions can be made.

ACKNOWLEDGMENT

This work has been carried out in the SOFIA ARTEMIS and GEMOM EU FP7 projects, funded by Tekes, VTT, and the European Commission.

REFERENCES

[1] D. M. Chess, C. C. Palmer, and S. R. White, "Security in an autonomic computing environment," *IBM Systems Journal, 42(1),* pp. 107-118. 2003.

[2] E. Ovaska, A. Evesti, K. Henttonen, M. Palviainen, and P. Aho, "Knowledge based quality-driven architecture design and evaluation," *Information and Software Technology, 52(6),* pp. 577-601. 2010.

[3] A. Evesti and S. Pantsar-Syväniemi, "Towards micro architecture for security adaptation," *1st International Workshop on Measurability of Security in Software Architectures (MeSSa 2010),* pp. 181-188, 2010.

[4] ISO/IEC 9126-1:2001. *Software Engineering - Product Quality - Part 1: Quality Model.* 2001.

[5] A. Avižienis, J. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing,* vol. 1, pp. 11-33, 2004.

[6] ISO/IEC 15408-1:2009, *Common Criteria for Information Technology Security Evaluation - Part 1: Introduction and General Model.* International Organization of Standardization, 2009.

[7] A. Herzog, N. Shahmehri, and C. Duma. (2009, "An ontology of information security," In *Techniques and Applications for Advanced Information Privacy and Security: Emerging Organizational, Ethical, and Human Issues,* Eds. H. R. Nemati, pp. 278-301, 2009.

[8] J. Zhou, "Knowledge Dichotomy and Semantic Knowledge Management," *Industrial Applications of Semantic Web,* pp. 305-316, 2005.

[9] C. Blanco, J. Lasheras, R. Valencia-García, E. Fernández-Medina, A. Toval, and M. Piattini, "A systematic review and comparison of security ontologies," *3rd International Conference on Availability, Security, and Reliability (ARES 2008),* pp. 813-820, 2008.

[10] A. Evesti, E. Ovaska, and R. Savola, "From security modelling to run-time security monitoring," *European Workshop on Security in Model Driven Architecture (SECMDA),* pp. 33-41, 2009.

[11] G. Denker, L. Kagal, and T. Finin, "Security in the Semantic Web using OWL," *Information Security Technical Report, 10(1),* pp. 51-58. 2005.

[12] A. Kim, J. Luo, and M. Kang, "Security Ontology for annotating resources," *LNCS,* vol. 3761, pp. 1483-1499, 2005.

[13] P. Savolainen, E. Niemelä, and R. Savola, "A taxonomy of information security for service centric systems," *33rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2007),* pp. 5-12, 2007.

[14] B. Tsoumas and D. Gritzalis. "Towards an Ontology-based Security Management," *20th Advanced Information Networking and Applications 2006 (AINA 2006),* pp. 985-992, 2006.

[15] F. García, M. F. Bertoa, C. Calero, A. Vallecillo, F. Ruíz, M. Piattini, and M. Genero, "Towards a consistent terminology for software measurement," *Information and Software Technology, 48(8),* pp. 631-644, 2006.

[16] N. F. Noy and D. L. McGuinness. "Ontology development 101: A guide to creating your first ontology,", pp. 1-25 2001.

[17] R. Savola, "A Security Metrics Taxonomization Model for Software-Intensive Systems," *Journal of Information Processing Systems, 5(4),* pp. 197-206, 2009.

[18] R. Savola and H. Abie. "Development of measurable security for a distributed messaging system," *International Journal on Advances in Security, 2(4),* pp. 358-380, 2010.

[19] C. Wang and W. A. Wulf, "Towards a Framework for Security Measurement," *Proceedings of the Twentieth National Information Systems Security Conference,* pp. 522-533, 1997.

[20] SPARQL Query Language for RDF, W3C Recommendation, http://www.w3.org/TR/rdf-sparql-query/, 31.1.2011

[21] A. Evesti and E. Ovaska, "Ontology-Based Security Adaptation at Run-Time," *4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO),* pp. 204-212, 2010.

[22] A. Evesti, M. Eteläperä, J. Kiljander, J. Kuusijärvi, A. Purhonen, and S. Stenudd, "Semantic Information Interoperability in Smart Spaces," *8th International Conference on Mobile and Ubiquitous Multimedia (MUM'09),* pp. 158-159, 2009.

[23] Semantic Information Interoperability in Smart Spaces, http://www.youtube.com/watch?v=EU9alk9t7dA, 31.1.2011

[24] J. Suomalainen, P. Hyttinen, and P. Tarvainen, "Secure information sharing between heterogeneous embedded devices," *1st International Workshop on Measurability of Security in Software Architectures (MeSSa 2010),* pp. 205-212, 2010.

[25] A. Elkhodary and J. Whittle, "A Survey of Approaches to Adaptive Application Security," *International Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2007 SEAMS '07.*, p. 16, 2007.

[26] A. Klenk, H. Niedermayer, M. Masekowsky, and G. Carle, "An architecture for autonomic security adaptation," *Ann Telecommun, 61(9-10),* pp. 1066-1082. 2006.

[27] C. J. Lamprecht and A. P. A. van Moorsel, "Adaptive SSL: Design, Implementation and Overhead Analysis," *First International Conference on Self-Adaptive and Self-Organizing Systems, 2007. SASO '07.*, pp. 289-294, 2007.

[28] C. J. Lamprecht and A. P. A. van Moorsel, "Runtime Security Adaptation Using Adaptive SSL," *Dependable Computing, 2008. PRDC '08. 14th IEEE Pacific Rim International Symposium,* pp. 305-312, 2008.

[29] G. Costa, F. Martinelli, P. Mori, C. Schaefer, and T. Walter, "Runtime monitoring for next generation Java ME platform," *Comput. Secur., 29(1),* pp. 74-87. 2010.

 ISBN: 978-1-61208-130-4

Involving Non Knowledge Base Experts With the Development of Ontologies

Vlad Nicolicin-Georgescu*, Vincent Bénatier
SP2 Solutions
La Roche Sur Yon, France
vladgeorgescun@sp2.fr, vbenatier@sp2.fr

Rémi Lehn, Henri Briand
*Ecole Polytechnique de l'Université de Nantes
Nantes, France
remi@fc.univ-nantes.fr, henri.briand@univ-nantes.fr

***Abstract* - The paper presents an approach to ontology development with the help of regular, non technical users. The specific objective is the construction of a software ontology with a much higher level of detail (e.g., patch version or software version compatibility) than existing propositions like OpenCyc. To this end, we need the feedback of software experts and users. The problem is that these are not knowledge experts with a background in working with ontology concepts, as required by the actual ontology development solutions. Our strategy is to provide an intuitive online platform through which users can provide feedback about their software configurations without the perquisites of ontology modeling. The platform, called *TimSys*, is linked with the ontology model via mapped data bases and it represents a bridge between the technical and non technical knowledge base worlds.**

Keywords – Ontology, Information System, Software, Decision Support System

I. INTRODUCTION

The evolution of *information systems* (IS) lead to complex description of their architectures, from hardware resources to installed software. As the number of software vendors increased exponentially, so did the number of offered functionalities and services. It is assessed that up to 90% of the requested functionalities is already available with existing applications [1]. The variety of software products implies an increased number of problems, from bugs to product incompatibility. These are referenced in different non or semi-structured sources, such as readme documents or technical forums. Integration propositions such as Microsoft's *knowledge base* (KB) articles for driver development [2] are very specific and are addressed to expert users.

In this context of problem resolution, whenever an issue occurs the user searches for answers with several sources, among which: the available documentation, call centers or technical forums and discussion lists. This way of functioning poses two major problems.

First, it requires a perfect knowledge of the used software configuration (vendor, name, version, patch, OS etc.). For example, if an interactive reporting software crashes constantly while using a specific data spreadsheet program, the user reports at best the reporting software version. He/she has no knowledge of the Java JRE version, which is actually the cause of the malfunction. As there is no complete description of the installed software, it will take several exchanges, e.g., with the support line, to determine that there is a third element at the root of the crash.

Second, each time an issue occurs, there is a repetitive and confusing process of software description. For instance, help desks employ three levels of competencies [3]. At each level, you are asked for your software configuration. If the problem isn't solved, each time you are in contact with a person from the help desk, you have to re-specify the software and the problem. This translates to frustrating repetitive operations and increased times for problem resolution. Moreover, it relates to the first problem as a non technical user is asked for detailed technical pieces of information.

With the expansion of the Semantic Web, ontologies have become a standard to model complex IS. Proposition of ontology usage for software models [4] or for semantic help desks [5] have proven that this may be a valid path to explore. Building and managing an ontology is not a trivial task, and is based on the collaboration of a specific user community. The problem is that this implies an expertise in working with ontologies and KBs, thus being reserved to a 'closed' category of users.

In consideration with the problems mentioned above, our objective is to build a software ontology, which should provide a reference point for system software description. For the initial ontology, we have chosen a restrained software perimeter, related to our expertise: *decision support systems* (DSS). To this end, we propose a semantic collaborative online platform, *TimSys*, which enables the description of user software environments starting from the ontology software concepts. The main idea behind *TimSys* is to help the evolution of the software ontology by integration of non technical user feedback. This way, everyone contributes to the development of the ontology, even if they are not KB experts.

The remainder of the paper is organized as follows. Section 2 presents the main software types with DSS and the problems of software configuration description. Section 3 shows how ontologies are used for knowledge modeling, and some of the advantages and drawbacks of using them. Section 4 introduces the *TimSys* platform, with the data

 ISBN: 978-1-61208-130-4

model and the use case scenarios. Finally, Section 5 sums up the conclusion and the future directions for this work.

II. SOFTWARE CONFIGURATION DESCRIPTION

First, this section introduces the main DSS software types, and then the existent problems and solutions with software description.

A. Decision Support Systems Software

DSSs represent the use case of our proposition. They are a type of IS that supports business and organizational decision-making activities. They have been thoroughly described by Inmon [6], with focus on data warehouse architectures. Software environments of DSSs include the following four major components:

(i) A data provider which contains the data that is integrated with the data warehouses. This data can be structured (e.g., DBs) or non-structured (e.g., technical documentation). The most often met solution is relational DBs (e.g., SQL Server, Oracle DB).

(ii) ETL (Extract, Transform, Load) software is in charge of transforming the provided data and loading it into the data warehouses. ETLs are usually developed by the data provider software editors (e.g., Oracle DW Builder, Data Integrator & Data Services by SAP).

(iii) The *data warehouse*, which stores the aggregated analytical data. Examples include the Oracle Hyperion Essbase or SAP Business Objects.

(iv) The *use interfaces* that provide access to the data from the data warehouse, usually for reporting (e.g., Hyperion Interactive Reporting , Microsoft Excel).

As decisional experts, we have been faced with the need of describing the software products above. Usually, the enterprises maintain this information in plain text documents, or eventually semi-structured ones (e.g., office documents with templates). This implies that every reference to the software configuration is based on a specific document, which must be provided each time. Moreover, version control has to be investigated for the documentation and for the software configuration. We have met several situations where software patches were applied without proper documentation (e.g. undocumented software migration). If the initial configuration specification is not updated, inconsistencies and false information occur.

B. Software Configuration Description

Software description offers many modeling alternatives. With the development of modeling tools such as UML or *Architecture Description Languages* (ADL), companies understood that integration and easy access are key factors for fast problem resolution.

In [7], the authors present an overview of the usage of UML with software architecture description. There is an extensive area of research over this subject, at a very detailed and technical level. Although they provide standardization with the description language, the complexity of such solutions is in most cases a ‘deal-breaker’ when facing simpler needs.

Another solution, less complex and simpler to use is system information software (e.g., Belarc Advisor [8]). For example, on Windows machines, the SOFTWARE registry keys contain reference to the installed software. Nevertheless, this solution has several drawbacks. First, it requires the installation of a specific program on each machine. Second, there is no complete view of the system (i.e., number of physical machines, how they are connected). Third, there is a problem with information availability, as the software list is not managed collaboratively; its sharing requires each time a duplication of the description file.

Our proposition is elaborated over the two modeling aspects presented above, taking the benefits of both. First, it uses a model complex enough, which enables the description of machines, software and the links that exist between them, but not too complex to enter the ADL world, while providing an intuitive interface for non-technical users. Second, by using ontologies, it overcomes the issues of availability and synchronization. Each software, configuration and system has its own unique URI, while assuring a complete system overview. Moreover, as the data model is opened, users benefit by adding feedback and continually improving it.

III. LINKED DATA AND ONTOLOGIES

With the development of the web and the expansion of the Internet, linked data is specified as the future of information throughout online environments. Developed by Berners-Lee, linked data is founded over the collaborative efforts of the Web 2.0 and the semantics of the future Web 3.0 [9]. The proposition states that the entire information on the web is part of a single global KB.

The formalization of the linked data concept is made through ontologies. Introduced by Gruber [10], an ontology defines a set of representational primitives able to model a domain knowledge or discourse [11]. An ontology allows the definition of three types of concepts: (i) classes (type of concept), (ii) individuals (instances of classes), and (iii) properties (links between classes and/or individuals). A sentence in an ontology is represented under the form of a triplet (subject, predicate, object), e.g., (Windows2003SRV, isA, Windows2003). Ontology expression languages are XML based, such as the W3C standards RDFS and OWL [12]. Additionally, SPARQL enhances SQL-like data query to retrieve information from ontologies.

Relating to the problem of software configuration, in [13], the authors provide an overview over how ontologies mix with UML. Moreover, some of our previous works with ontology models for managing DSSs [14] have shown the advantages and inconveniences of ontology modeling.

The benefits of using ontologies come from the dynamics of the data model, high expressivity and inference support. Dynamics refers to the fact that the information model is prone to constant changes (unlike DBs implementations), as collaboration is the key to building an ontology. High expressivity indicates that any matter or facts can be expressed within the ontology (from where the three levels of expressivity with the OWL). Last, inference allows the deduction of new knowledge from the existing knowledge by using axioms and rules.

 ISBN: 978-1-61208-130-4

On the contrary, the main drawbacks of ontologies are the novelty of the technology, information retrieval performance and high technical competences requirement. Only recently the industry has shown its interest towards this technology (i.e., Oracle 10g semantic module). Data retrieval performance for large scale ontologies proves to be a problematic point, from where the recommendation that for high number of concepts, relational DBs are preferred for faster access [15]. The last major inconvenient is that working with ontologies (as with any new technology) requires a formal technical preparation. As ontologies aim to sustain a large collaborative online usage, this sends aback a good part of its 'target' users.

Our proposition is implemented with regards to these disadvantages. We use a combined data model (DB + OWL), to assure high expressivity and collaboration, while providing fast data access.

IV. TIMSYS

TimSys [16] is our proposition for a collaborative, semantic, online and non-technical software configuration description platform. Collaborative expresses the fact that its users play an active role in the evolution of the software KB. Semantic indicates the usage of ontologies for KB formalization. Online sends to the standardization of the software concepts. Last, non-technical underlines that anyone can contribute to the development of the KB, regardless his background in working with ontologies.

TimSys is composed of two main modules: (i) the software and systems KB and (ii) the user interface .

A. The Knowledge Base

The KB contains the totality of available software, configurations and systems. As mentioned earlier, the data model is a combination between relational DBs and OWL ontologies.

OWL is used to implement the software ontology with a very high granularity. All the details from editor to patch version are described. Additionally, it allows the dynamic description of relations between software such as compatibility or functionalities. Inference rules play a very important role, as to establish software version dependencies. Our prototype software ontology contains 902 individuals, 25 classes, 13 object properties and 14 data type properties with the OWL DL expressivity.

An example of a software concept from the ontology (Windows Server 2003 R2) is shown in the following table

Table 1 – Software ontology concept example (triple)

Subject	***Predicate***	***Object***
Win2k3SRV_R2	rdf:type	Win2k3SRV
	hasMajorVersion	"5"^^xsd:string
	hasVersion	"5.2"^^xsd:string
	isCompatibleWith	Essbase_9.2.1
	hasPrevious	Win2k3SRV_SP1
	hasEditor	Microsoft

First, we notice the inclusion of the *Win2k3SRV_R2* individual the general *Windows2k3SRV* class via the *rdf:type* property. Then, two string data properties indicate the version of the software. The object property *isCompatibleWith* specifies the list of software with which it is compatible. The *hasPrevious* property links this specific version with the Win2k3 server products timeline. Last, the *hasEditor* property links it with the *Microsoft* concept, which is obtained by querying the DBPedia SPARQL endpoint [17] as an already existing concept.

Relational DBs are used for the representation of the systems and the corresponding software configurations. We define a *configuration* as the totality of (interesting) software installed on a single machine (the OS and the installed software). A *system* is defined as a combination of one or several configurations; which we consequently call a *timsys*.

The choice of DBs for the implementation comes from the fact that, unlike the software ontology (where there is a reduced number of concepts), the number of configurations and systems may reach billions. In order for the DB model to be as fast as possible, a mapping of the software ontology concepts is done such that they are also formalized in the DB model. This means that the data backend is fully assured by relational DBs. Any evolution in the ontology model results in its remapping into the DB model. This permits a constant availability of the software lists, while new changes to the software ontology are processed.

B. Evoloving the Ontology - The User Interface

The graphical user interface is the front end of the TimSys platform. It includes two major different sub interfaces: (i) access and (ii) management.

The access interface enables the online view of a system by an unique hash link (e.g., http://www.timsys.org/hjJKuyJ8). Using this link, anyone at anytime can have a look at the configurations and list of software. This greatly helps the problematic of ambiguity, duplication and synchronization while offering access to an opened software KB.

The management administration interface proposes the creation of new systems or the modification of existent ones, similarly via an unique link. The interface is build with regards to non-technical users, thus remaining as simple as possible. Alongside permitting users to describe their configurations, it provides the possibility of feedback in the cases where a required software is not found in the KB.

This is the *key aspect* towards the evolution of the software ontology. Whenever a software is not found, the user fills a three field form with the software editor, name and version (only the last 2 are compulsory). This feedback is stored temporarily in the DB, becoming instantly available to the user and with the destination of ontology integration. At this point, the KB experts are in charge of updating the software ontology accordingly. Once the ontology updated (e.g., once per day), the remapping of the new concepts to the DBs is made, and the new software becomes permanently available. We note that the intervention of a KB expert is always needed for integration.

Summing up, Figure 1 shows the general functioning of the *TimSys* platform, with the presented modules and use cases.

 ISBN: 978-1-61208-130-4

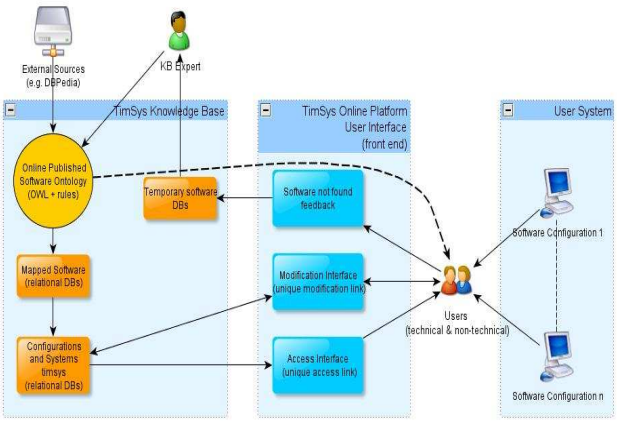

Figure 1 – *TimSys* Overall Architecture

The arrows indicate the direction in which the data flows. We notice that this is both ways for the management modification interface. Moreover, the ontology is published online and available to users for direct access of its concepts.

V. CONCLUSIONS

In this paper, we presented a proposition for building an ontology by involvement of non KB experts. Specifically, the objective was to develop a software ontology by integrating a maximum of user feedback. To this end, we proposed an online platform for describing software configurations. Consequently, we proposed solutions to overcome the issues of collaboration, synchronization and availability when it comes to describing the software environments, with the use case of DSSs.

The state of the art offered a view over software configuration modeling approaches and over the semantic web technologies. With our proposition, *TimSys*, we have presented a combined DB/OWL data model and an intuitive interface for access and management. Thus, we have seen how non-technical users feedback contributes to the development of the software ontology.

Nevertheless, the work presented here is at an early stage. Our future works will detail the aspects of: DB/OWL mapping with the data model, usage of existing software KB (more than DPBedia), validation of the ontology and its publication as a recognized reference. As it is an open source project, we aim at building an active community around *TimSys*, for both technical and non-technical feedback.

REFERENCES

[1] P. Oreizy, “Decentralized software evolution,” in The International Conference on the Principles of Software Evolution (IWPSE 1), 1998. Last accessed November 2010. Available: http://www.ics.uci.edu/~peymano/-papers/iwpse98/

[2] Microsoft. Knowledge base articles for driver development. Last accessed November 2010. Available: http://www.microsoft.com/whdc/driver/-kernel/kb-drv.mspx

[3] T. N. I. A. System. Help desk level competencies. Last accessed November 2010. Available: http://-www.nitas.us/docs/-Help%20Desk%20Level%20Competencies.pdf

[4] M. Brauer and H. Lochmann, “An ontology for software models and its practical implications for semantic web reasoning,” in The 5th European semantic web conference on The semantic web: research and applications, ESWC’08, 2008.

[5] Nepomuk. Mandriva community case study first prototype of a social semantic help desk. Last accessed November 2010. Available: http://-nepomuk.semanticdesktop.org/xwiki/bin/download/-Main1/D11-2/-D11.2_v10_NEPOMUK_1st%20Prototype%20of%20Social%20Semantic%20Helpdesk.pdf

[6] W. H. Inmon, Building the data warehouse, fourth edition, W. Publishing, Ed. Wiley Publishing, 2005.

[7] P. Avgeriou, N. Guelfi, and N. Medvidovic, “Software architecture description and uml,” UML MODELING LANGUAGES AND APPLICATIONS, vol. 3297, pp. 23–32, 2005.

[8] Belarc. The belarc advisor. Last accessed January 2011. Available: http://www.belarc.com/free_download.html

[9] C. Bizer, T. Heath, and T. Berners-Lee, “Linked data - the story so far,” International Journal on Semantic Web and Information Systems (IJSWIS), vol. 5, no. 3, pp. 1–22, 2009.

[10] T. Gruber, What is an ontology? Academic Press Pub., 1992.

[11] L. Liu and M. T. Özsu, Encyclopedia of Database Systems, L. Liu and M. T. Özsu, Eds. Springer-Verlag, 2008. Available: http://tomgruber.org/writing/ontology-definition-2007.htm

[12] W3C. World Wide Web consortium. W3C. Last accessed July 2010. Available: http://www.w3.org/

[13] J. Savolainen, “The role of ontology in software architecture,” in OOPSLA Workshop on How to Use Ontologies and Modularization to Explicitly Describe the Concept Model of a Software Systems Architecture, 2003.

[14] V. Nicolicin-Georgescu, V. Benatier, R. Lehn, and H. Briand, “Ontology-based autonomic computing for decision support systems management,” in The First International Conference on Models and Ontology-based Design of Protocols, Architectures and Services, MOPAS 2010, pp. 233–236, 2010

[15] N. K. Irina Astrova and A. Kalja, “Storing owl ontologies in sql relational database,” International Journal of Electrical, Computer, and Systems Engineering, vol. 1, no. 4, pp. 242–247, 2007.

[16] This is my System (TimSys). Last accessed January 2011. Available: www.timsys.org

[17] DBpedia. DBpedia SPARQL endpoint. Last accessed January 2011. Available: http://dbpedia.org/sparql

 ISBN: 978-1-61208-130-4

Formal Logic Based Configuration Modeling and Verification for Dynamic Component Systems

Zoltan Theisz
evopro Informatics and Automation Ltd.
Email: zoltan.theisz@evopro.hu

Gabor Batori
Software Engineering Group
Ericsson Hungary
Email: gabor.batori@ericsson.com

Domonkos Asztalos
Software Engineering Group,
Ericsson Hungary
Email: domonkos.asztalos@ericsson.com

***Abstract*—Reconfigurable networked systems have often been developed via dynamically deployed software components that are executing on top of interconnected heterogenous hardware nodes. The challenges resulting from the complexity of those systems have been traditionally mitigated by creative ad-hoc solutions supported by domain specific modeling frameworks and methodologies. Targeting that deficiency, our paper shows that by involving a first-order logic based structural modeling language, Alloy, in the analysis of component deployment we could extend the limits of the generic domain specific metamodeling methodology developed for Reconfigurable Ubiquitous Networked Embedded Systems.**

***Keywords*-Alloy specification; formal model semantics; metamodeling; dynamic component system**

I. INTRODUCTION

Reconfigurable networked component systems provide a versatile platform for implementing highly distributed autonomic peer-to-peer applications in domains of both real sensor networks and autonomic computing [1] environments such as e.g. intelligent network management that relies on sensory and effectory facilities of multi-level control loops. The building and verification of those applications in practice has turned out to be a rather challenging research topic that could enormously benefit from the usage of domain specific modeling approaches. One of the major results of the Reconfigurable Ubiquitous Networked Embedded Systems (RUNES) IST project was to establish a reflective distributed component-based multi-platform middleware architecture [2] for heterogeneous networks of computational nodes, including metamodel-based software development methodology [3] and graphical development framework. The RUNES metamodel provides all the relevant concepts software architects need to efficiently utilize the computational resources within a reflective distributed component-based environment. Due to the inherent complexity of distributed reconfigurable component systems, we advocate the usage of Alloy [4], a formal first order logic based language supported by a fully automated analyzer that has been successfully used to model various complex systems in a wide range of application domains for domain specific model verification purposes. Alloy has been applied in [5] for the analysis of some critical correctness properties that should be satisfied by any secure multicast protocol. The idea of using Alloy for component based system analysis was suggested by Warren et al. [6]. This paper shows OpenRec, a framework which comprises a reflective component model and the Alloy model of OpenRec. This Alloy model served as a basis for our Alloy component model but our model is more detailed which enables deeper analysis of the system behavior. Moreover, [7] demonstrates an Alloy model that identifies the various types of dynamic system reconfigurations. It provides a good categorization of various problems and solutions related to dynamic software evolution. Furthermore, Aydal et al. [8] found Alloy Analyzer one of the best analysis tool for state-based modeling languages.

Although individual application scenarios can be easily expressed manually in Alloy we firmly believe that the synergy between metamodel driven design and first order logic based practical model verification could result in a more advantageous unified approach. Our approach, in a nutshell, semi-automatically generates all the relevant RUNES deployment configuration assets that have also been analyzed within Alloy. By analyzing a significant subset of frequently reoccurring configurations the boundary between valid and invalid component configurations can be thoroughly investigated against proper sets of model-based application and/or middleware feasibility constraints. The analysis results can be used to provide input to the runtime adaptive control logic in order to extend the model-based software development framework [3] with effective autonomicity.

In this paper, we will describe how a first-order logic based model of the RUNES middleware has been developed in Alloy and how it has been integrated into the RUNES domain specific modeling framework and methodology [3]. In the remainder of the paper, first in Section II, we briefly overview Alloy, then, we also disseminate in detail how the RUNES Metamodel has been formalized in it. Next, Section III explains how the Alloy backed verification step gets integrated into the general metamodel-based RUNES application development methodology. Then, Section IV presents a short introduction into the usage how verification results can be incorporated into a full scale model-based

 ISBN: 978-1-61208-130-4

management approach. Next, in Section V, a simplified real-life sensor application will be showcased in order to visualize our approach via a tangible example. Finally, in Section VI we conclude the paper and briefly highlight our future research plans.

II. RUNES METAMODEL VERIFICATION WITH ALLOY

A. Alloy

Alloy [4] is a textual metamodeling language that is based on structured first-order relational logic. A particular model in Alloy contains a number of signature definitions with fields, facts, functions and predicates. Alloy is supported by a fully automated constraint solver, called Alloy Analyzer [9], which can be used to verify model parameters by searching for either valid or invalid instances of the model.

B. Applying Alloy for component system verification

The RUNES reflective component middleware has been created as one of the most important software assets resulting from the RUNES IST project. In general, it consists of a component-based middleware that follows the currently popular loosely-coupled paradigm of Service Oriented Architecture [10]. The middleware is fully reflective, its API is rigorously specified in various programming languages [11], its elements are conceptually backed by multi-layer metamodels and finally a metamodel driven, domain specific application development methodology provides the guidelines for its most effective application. All in all, the RUNES application development is highly streamlined and it is carried out mostly following strict model-based design principles within a semi-automatic metamodeling environment. Although the development techniques and the guiding process have been streamlined, the verification and the validation of the resulting modeling assets such as the application models and the incorporated executable action semantics have to be carried out manually or semi-automatically [12]. Model based test generation proved to be very effective in some scenarios, though testing cannot replace, even in industrial setups, the verification and/or validation efforts. Our industry experience has also proved quite frequently that modeling tasks are highly creative, thought intensive activities which result in complex artifacts of great variability. Therefore, the verification and validation of model-based solutions is a considerable challenge. Fortunately, in reconfigurable sensor applications, the complexity of the resulting system is slightly limited because the variability of the system mainly originates only from the flexibility of the underlying component model of the middleware, thus a more formal way of verifying applications is within practical reach. From the wide spectrum of verification paradigms first-order logic is considered as one of the most rigorous approaches. It has turned out that in our scenarios mostly the independently acting sub-configurations of tightly coupled components have usually caused the majority of the most

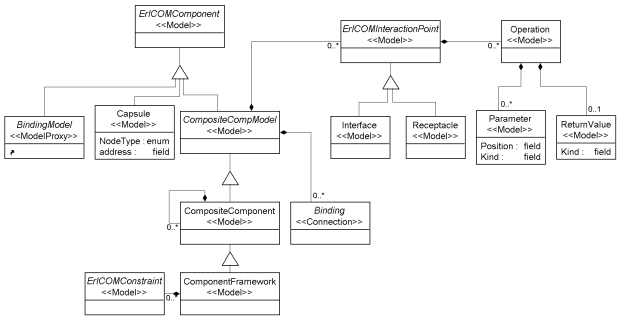

Figure 1. Kernel part of the metamodel

serious malfunctions; therefore, our aim had been mainly directed towards their automatic elimination and avoidance in the runtime deployment. The generic design principles of Alloy [4] facilitates both easy meta-language creation that complies with metamodel driven domain specific language building concepts and formal verification of models. In the following, we will formalize the semantics - from the verification point of view - of our middleware metamodel in Alloy in full accordance with the principle of the semantic anchoring approach reported in [13].

C. Functional metamodel

The RUNES Metamodel specifies the formal metamodel that represents the relevant elements of the RUNES middleware architecture [2]. Figure 1 illustrates the kernel part of the metamodel, which defines the basic concepts of Interfaces, Receptacles, Components and Bindings including their relations and cardinalities. Combined with the associated OCL expressions the RUNES Metamodel establishes a model-based application development environment in GME [14], which enables rapid RUNES application development. In order to be able to verify the proper configuration sequence of a particular modeled application scenario the RUNES Metamodel has to be semantically anchored to a precise structural and behavioral formalism in Alloy. Therefore, the following paragraphs will show how the various metamodeling concepts have been reformulated in Alloy so that application models could be verified against configuration constraints.

In general, the functional specification of any RUNES application must be organized around Components and Bindings. The Components represent the encapsulated units of functionality and deployment. The interactions amongst them take place exclusively via explicitly defined Interfaces and Receptacles. The dynamic behavior of the components are automatically generated from Message Sequence Charts (MSC) and the results are formalized via concurrent Finite State Machines (FSM) [15]. Therefore, a generic RUNES Component is defined as a signature whose fields consist of at most one state machine and a set of Interfaces and Receptacles, respectively.

 ISBN: 978-1-61208-130-4

```
abstract sig Comp{
   state_machine:set StateMachine,
   provided: set Interface,
   required: set Receptacle,
}{
   lone state_machine
}
```

Both the Interface and the Receptacle inherit the common characteristics of an Interaction Point, which is defined by a set of related operation signatures and associated data types. The Interface represents the "provided", the Receptacle the "required" end of a component-to-component connection, respectively.

```
abstract sig Signature{}
abstract sig InteractionPoint {
   signatures: set Signature
}
sig Interface extends InteractionPoint{}
sig Receptacle extends InteractionPoint{}
```

Bindings ensure that connections between Interfaces and Receptacles are set up consistently, according to their proper definitions. Hence, a Binding is defined as a signature that contains fields for one Interface and one Receptacle and one non-identical, component correct mapping that connects the previously mentioned two items.

```
abstract sig Binding{
   mapping:Comp -> Comp,
   interface: one Interface,
   receptacle: one Receptacle
}{
   one mapping
   no (mapping & iden)
   receptacle in (Comp.~(mapping)).required
   interface in (Comp.mapping).provided
}
```

The Receptacle must always represent a requirement which is a 'subset' of the operations (signatures) provided by the Interface it intends to be bound to via the Binding. That fact has to be made explicit in Alloy to allow only correct Bindings in the model.

```
all b:Binding| b.receptacle.signatures in b.interface.signatures
```

D. Deployment metamodel

Figure 2 shows those relevant deployment concepts of the RUNES Metamodel that determine the runtime aspects of a RUNES component application. The key element of the metamodel is the Capsule that represents the generic middleware container, which on the one hand provides direct access to all the functionalities of the runtime API [11], on the other hand it manages a robust fault recovery and redundancy facility. Deploying a component into a capsule in generic terms means that it must be ensured that adequate resources are available for loading in the component in a particular instance of time. The deployed components and bindings might change in time, hence their temporal representation must take into account the explicit definition of Time, too. The Capsule also possesses a distributed, peer-to-peer, fully reflective meta-data repository that can be used for both application and middleware specific purposes. A Capsule in Alloy is defined as a signature having fields representing the temporal evolution of deployed components, bindings and a middleware related resource pool plus the time invariant capsule topology information.

```
open util/ordering[Time] as TO
sig Time{}
abstract sig Capsule {
```

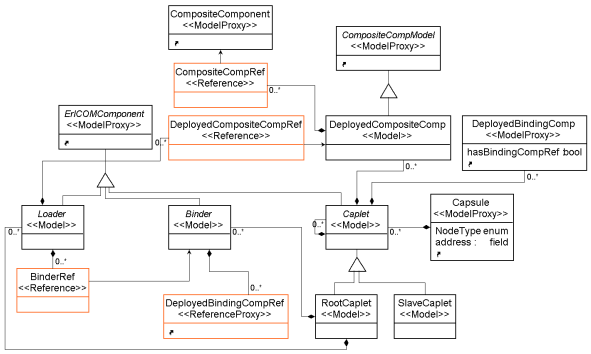

Figure 2. Deployment part of the RUNES Metamodel

```
   comps: DeployedComp -> Time,
   bindings: DeployedBinding -> Time,
   comp_capacity: Int -> Time,
   neighbours: some Capsule
}{
   all t:Time|int[comp_capacity.t] >= #(comps.t)
   all t:Time|comp_capacity.t >= Int[0]
}
```

A deployed component incorporates all the necessary information that stores the active process aspect of the component's functionality including explicit definition of state transitions in time. In other words, the deployed component can be considered as a dynamic instance of a component in accordance with its "ModelProxy" declaration in GME depicted in Figure 2. The temporal aspect of the state transitions are defined by the fire and the current_state fields of the DeployedComp signature.

```
sig DeployedComp{
   deploy: one Comp,
   fire: Transition -> Time,
   current_state: State -> Time
}{
   deploy in FunctionalConf.comps
   all t:Time|lone fire.t
   all t:Time|lone current_state.t
}
```

A deployed binding does not declare time explicitly, however, it contains a mapping field between the two participating deployed components, hence, it is also time dependent. The compatibility of the deployed binding is checked based on the functional definition of the connection.

```
sig DeployedBinding{
   mapping: DeployedComp -> DeployedComp,
   deploy: one Binding
}{
   one mapping
   (DeployedComp.~mapping).deploy =
     Comp.~(deploy.mapping)
   (DeployedComp.mapping).deploy = Comp.(deploy.mapping)
}
```

Our main goal of applying Alloy has been oriented towards configuration verification, thus we must represent a deployed RUNES application in Alloy as a collection of capsules which register the temporal evolution of each of the components and the bindings. Alloy's trace statements help us verify the time evolution of the application against feasibility constraints and the successful runs can also be visualized for human inspection, too.

```
sig DeploymentConf{
   capsules: some Capsule
}
```

The RUNES middleware API supports a set of component management operations such as [un]loading, [un]binding

 ISBN: 978-1-61208-130-4

and migrating components. The operations require time to execute their functionality, they usually modify only local states of the distributed application and keep the rest unchanged. In Alloy, we serialize the potentially concurrent atomic API operations in such a way that one and only one of them can be carried out in one particular instance of time. Due to size constraints only the definition of the migrate operation is presented here in detail. The other operations have been defined applying similar specification techniques.

Component migration is carried out between two capsules by moving an already deployed component between two consecutive points of time. First the preconditions are checked if it is a real migration between two different capsules and there are enough resources available in the receiving capsule. Then, local states of the two respected capsules are to be updated and, finally, three constraints are to be satisfied so that the rest of the application state remains unchanged.

```
pred migrate(c_src,c_dst:Capsule,d:DeployedComp,t,t':Time){
   c_src != c_dst
   #(c_dst.comps.t) < int[c_dst.comp_capacity.t]
   c_dst.comps.t' = c_dst.comps.t+d
   c_src.comps.t' = c_src.comps.t-d
   all capsule:Capsule|capsule.bindings.t'=capsule.bindings.t
   all capsule:Capsule-c_src-c_dst| capsule.comps.t'=capsule.comps.t
   all capsule:Capsule| capsule.comp_capacity.t' = capsule.comp_capacity.t
}
```

Above all those previous definitions, the RUNES Metamodel also enforces a couple of RUNES specific restrictions over the possible component configurations in order to safeguard that only semantically correct component reconfigurations are permitted. In the metamodel (see Figure 1 and Figure 2) those rules are expressed either via cardinality constraints or via OCL expressions. Therefore, the Alloy formalism must incorporate the corresponding definitions, too. Here only the most important elements of that constraint set are summarized.

- A Binding or a Component must be contained within at most one single Capsule

```
no disj capsule1,capsule2:Capsule|
 some (capsule1.bindings) & (capsule2.bindings)
no disj capsule1,capsule2:Capsule|
 some (capsule1.comps) & (capsule2.comps)
```

- Two Bindings of the same type must not be deployed if they share the same Receptacle.

```
no disj b1, b2:DeployedBinding| (b1.deploy = b2.deploy)
 and (b1.mapping.DeployedComp = b1.mapping.DeployedComp)
```

- There must not be such a Binding within a Capsule that has a connected Component which is not deployed in any of the Capsules

```
no deployedBinding:DeployedBinding|some t:Time|
 deployedBinding in Capsule.bindings.t and
 (deployedBinding.mapping.DeployedComp not in Capsule.comps.t
 or deployedBinding.mapping[DeployedComp] not in Capsule.comps.t)
```

E. Behavior metamodel

The internal dynamics of the components' functional behavior is modeled in Alloy by an explicitly specified Finite State Machine (FSM) that takes into account all changes of the internal state of vital components, the conditionality of state transitions and the necessary action semantics required when a new state has been entered. Our FSM definition in Alloy mirrors the formal mathematical model following the generic principle of semantic anchoring [13].

```
abstract sig State{}
abstract sig Transition{
   trans: State -> State
}{
   one trans
}
abstract sig StartState extends State{}
abstract sig StartTransition extends Transition{}
pred transition[d:DeployedComp,t,t':Time]{
   (d.fire.t).trans.State = d.current_state.t
   (d.fire.t).trans[State] = d.current_state.t'
}
abstract sig StateMachine{
   states: some State,
   startState: one StartState,
   transitions: some Transition,
   startTransition: one StartTransition,
}{
   no (states & startState)
   no (transitions & startTransition)
}
fact Traces{
   ...
   all t:Time-TO/last[],d:DeployedComp|let t'=TO/next[t]|
     some d.fire.t => (transition[d,t,t'])
   all t:Time|some DeployedComp.fire.t
}
```

To round up the section, Figure 3 depicts a concrete model that instantiates the above introduced RUNES Metamodel in Alloy. It shows a snapshot from a dynamicly evolving component configuration of a sensor network scenario where the components have been deployed over a cross shaped capsule topology - indicated by green arrows - within which the resource pools are also capacity limited. The mapping of the components and bindings onto the capsules in a particular instance of time is visualized by the brown and red arrows respectively.

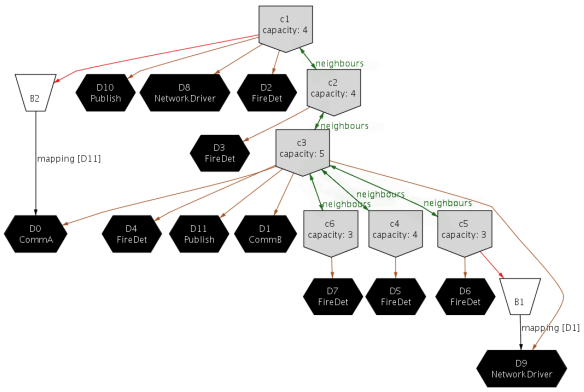

Figure 3. Scenario analysis snapshot

III. PROCESS

The RUNES application development process has a well defined five-layer architecture [3] that guides the application developer through the Scenario, the Application Modeling, the Platform, the Code Repository and the Running System stages. The presented Alloy based model verification approach builds on a first-order logic based formalism, which extends the RUNES development process. As Figure 4 shows, the extension has been realized by two additional model transformations that turn RUNES Component Models and corresponding RUNES Deployment Models into configuration scenarios that can be verified within the Alloy

 ISBN: 978-1-61208-130-4

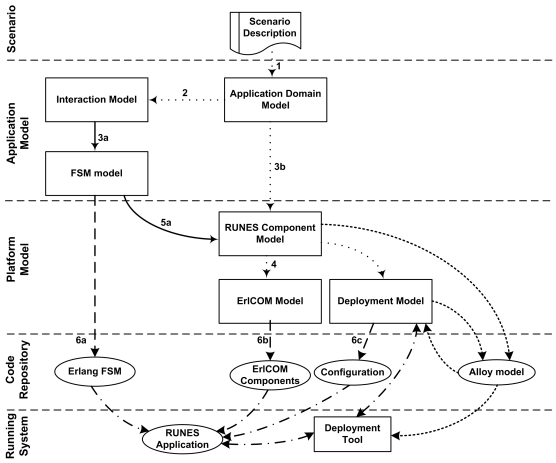

Figure 4. Software Development Process extended with Alloy verification

Analyzer. The model transformations produce configuration scenarios which contain both structural and behavioral specifications of the application. However, only those parts of the FSM action semantics are kept from the total dynamic behavior which directly relate to the internal control logic of the scenario. These parts precisely specify when and with which parameters the application invokes the runtime APIs provided by the RUNES middleware.

The verification of a particular scenario investigates the evolution of the application from the point of view of the component reconfigurations enabled by the RUNES middleware which are mainly restricted by the resource availability within the capsules along the time. The results of the verification provide input to the runtime autonomic control mechanisms that manage pre-calculated adaptive component reconfiguration. The approach is usually iterative and the convergence criteria are decided on a case-by-case basis.

IV. METAMODEL-DRIVEN COMPONENT MANAGEMENT

Metamodel-driven component management is an interesting new way of generalizing policy-based network management [16] in such a way that the information model used by the network management infrastructure mirrors those software assets of the component based system that are produced by the model translators. In effect, the model based system design is kept intact and extended by elaborated action semantics. From the point of view of model-based application control, the most important element in the RUNES runtime architecture is the Deployment Tool, which establishes a soft real-time synchronization loop between the GME model repository and the running component application. The schematics of the Deployment Tool based reconfigurability is shown in Figure 4. The Deployment Tool, a protocol independent abstraction of GANA's Decision Making Element [17], first deploys the initial component configuration of the application then it constantly readapts the component configuration by listening to both application and middleware notifications and by continuously re-evaluating the configuration in hand. The core of the control logic is based on the verification results from previous Alloy analyzes. Moreover, it visualizes the actual component configuration of the system in a metamodel compliant view within GME and also takes indirect corrective actions by modifying the resource availability of the capsules via RUNES middleware API invocations. Currently, the control logic is not automatically generated from a batch of Alloy verifications; however our aim is to adopt the GANA [17] control meta-model and to populate it via an automatic model transformation directly from the instantiated RUNES metamodel in the Alloy verification phase. With the control logic properly established, the Deployment Tool is capable to function both as a re-active or a pro-active component reconfigurator as reported in [3], [18].

V. SIMPLIFIED SCENARIO EXAMPLE

In this section, a simplified example will demonstrate how Alloy helps the model verification. For the sake of easy comprehension, here, only a simplified configuration example has been chosen, which incorporates merely two capsules and 8 deployed components. Although this logic based approach, in general, is rather resource intensive, realistic scenarios by a magnitude larger in size are still possible to be analyzed successfully in this manner. Nevertheless, large reconfiguration setups must be optimized individually; therefore our current approach is, in virtue, semi-automatic.

In Figure 5, the Alloy representation of the functional configuration of the component system is depicted.

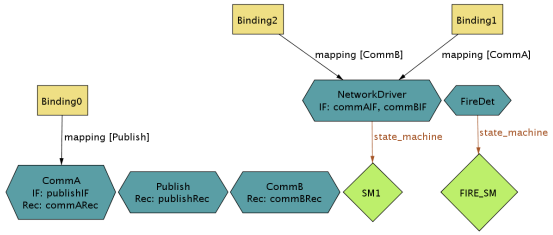

Figure 5. Functional configuration of the example system

The functional view of the investigated system contains five different component types, namely, the network related three components (NetworkDriver, CommA and CommB) and the two application specific components (Publish and FireDet). CommA and CommB implement two different communication paradigms relying on the functionality of the common NetworkDriver component through Binding1 and Binding2. The Publish component's main functionality is to broadcast different sensory measurement data towards the processing end points. The FireDet component is the control component which reconfigures the other components

 ISBN: 978-1-61208-130-4

whenever it has detected fire situation. The main goal of the reconfiguration is to keep the sensor system in operation even in case of extreme fire conditions. The reconfiguration is carried out by migrating the application functionalities to other capsules, which are located in the neighborhood. By decreasing the generic capacity parameter of the current capsule other capsules will not be able to immediately push back newly migrated components. Both the NetworkDriver and the FireDet components possess proper state machines which are represented by the diamonds in Figure 5.

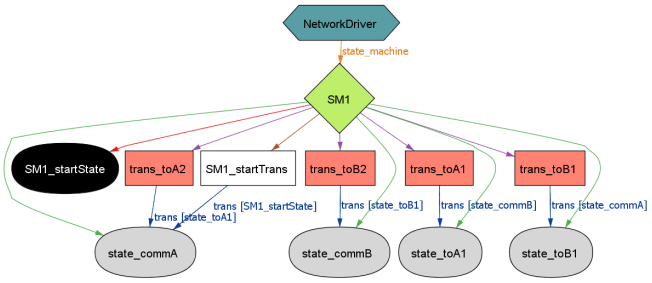

Figure 6. NetworkDriver component state machine

Figure 6 shows the state machine of the NetworkDriver component. The black ellipse shows the start state, while the white rectangle represents the transition from the start state to another state, which is called state_commA in this particular case. Via the transition from state_commA to state_commB, through a temporal state_toB1, the unbinding of component CommA from NetworkDriver and the binding of component CommB to NetworkDriver take place. This state change clearly represents the reconfiguration of the communication paradigm.

Figures 7–10 show an Alloy trace sequence. The resulting model is projected over Time in such a way that the relations rooted in Time are represented through a sequence of models. More precisely, one Time instance is connected to one particular Model snapshot.

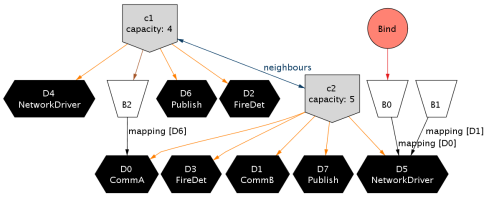

Figure 7. Component binding step

Figure 7 presents the first step of the sequence. When SM1_startTrans is activated the circle with the Bind tag points to the deployed binding B0. The deployed component D0 and D5 will be bound in the following step (see Figure 8).

In Figure 8 the first reconfiguration of the system can be seen. The FireDet component's state machine is activated,

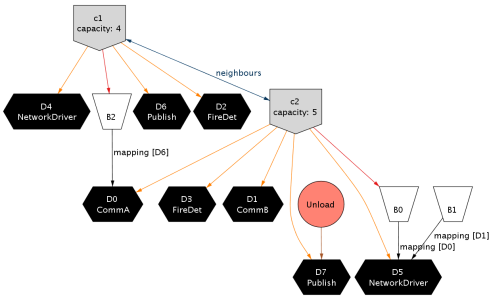

Figure 8. Component reconfiguration (unload) step

hence the migration of the application functionality has been started. Since the Publish component has been deployed to the neighboring capsule, the FireDet component, instead of migrating the marked component, is going to unload the Publish component from the second capsule. Furthermore, it will decrease the capacity of the capsule.

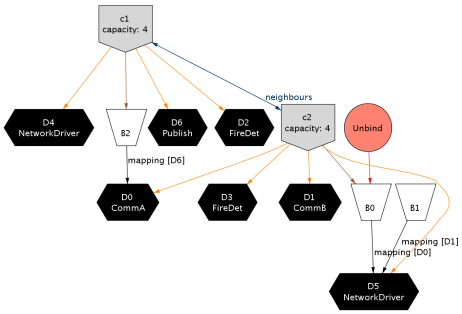

Figure 9. Component unbinding step

In Figure 9, the reconfiguration of the NetworkDriver component from CommA to CommB has started. In Figure 10, the second migration attempt is demonstrated. In this case, component CommA is migrating to the first capsule because this required functionality has not been deployed to that capsule so far.

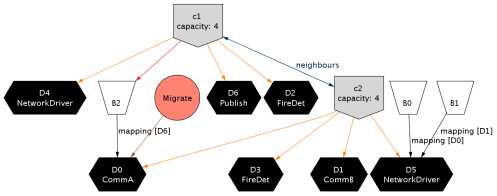

Figure 10. Component reconfiguration (migration) step

This simplified example indicates the way how a particular verification session takes place using the Alloy Analyzer. It helps generate configuration sequences which comply with application constraints. The current verification approach mainly focuses on the problem domain of component reconfigurability; thus, it assists the run-time control logic by

 ISBN: 978-1-61208-130-4

identifying situations with serious capacity limitations of the deployed capsules.

VI. Conclusion

This paper presents a new way of combining domain specific metamodeling techniques with first-order logic based metamodel verification so that model building could facilitate later run-time control mechanisms of the modeled system. We have introduced the semantical foundations of our approach and detailed its applicability in the case of reconfigurable component based sensor networks. A simplified example has been disseminated to illustrate the benefits of the approach. Our current work is to combine the RUNES meta-model and the GANA meta-model and to automate the generation of the adaptive control logic, based on the verification of the model based component configurations, to manage the deployed system. We are aware of the scalability issues of our approach, so further studies will be carried out in this regard. Moreover, the results of these studies will get incorporated, as best practices guidelines, into model translators that are supposed to produce the majority of the Alloy specifications. Ultimately, our aim is to create a generic framework which iteratively and interactively modifies and verifies the component model of sensor application scenarios and continuously indicates the most probable correct run-time configuration sequences thereof.

References

[1] "An architectural blueprint for autonomic computing." *Autonomic Computing*, IBM White Paper, June 2005.

[2] P. Costa, G. Coulson, C. Mascolo, G. P. Picco, and S. Zachariadis, "The runes middleware: A reconfigurable component-based approach to networked embedded systems," *Proc. of the 16th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC'05), Berlin, Germany*, September 2005.

[3] G. Batori, Z. Theisz, and D. Asztalos, "Domain specific modeling methodology for reconfigurable networked systems," *ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2007)*, 2007.

[4] D. Jackson, *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, London, England, 2006.

[5] M. Taghdiri and D. Jackson, "A lightweight formal analysis of a multicast key man-agement scheme," *Formal Techniques for Networked and Distributed Systems (FORTE 2003)*, vol. 2767 of LNCS., pp. 240–256, 2003.

[6] I. Warren, J. Sun, S. Krishnamohan, and T. Weerasinghe, "An automated formal approach to managing dynamic reconfiguration," *21st IEEE International Conference on Automated Software Engineering (ASE 2006), Tokyo, Japan*, pp. 37–46, September 2006.

[7] D. Walsh, F. Bordeleau, and B. Selic, "A domain model for dynamic system reconfiguration," *ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2005)*, vol. 3713/2005, pp. 553–567, October 2005.

[8] E. G. Aydal, M. Utting, and J. Woodcock, "A comparison of state-based modelling tools for model validation," *Tools 2008*, June 2008.

[9] D. Jackson, "Alloy analyzer," *http://alloy.mit.edu/*, 2008.

[10] T. Erl, "Soa principles of service design," *Prentice Hall*, 2007.

[11] G. Batori, Z. Theisz, and D. Asztalos, "Robust reconfigurable erlang component system," *Erlang User Conference, Stockholm, Sweden*, 2005.

[12] G.Batori and D. Asztalos, "Using ttcn-3 for testing platform independent models," *TestCom 2005, Lecture Notes in Computer Science (LNCS) 3502*, May 2005.

[13] K. Chen, J. Sztipanovits, S. Abdelwahed, and E. Jackson, "Semantic anchoring with model transformations," *European Conference on Model Driven Architecture -Foundations and Applications (ECMDA-FA), Nuremberg, Germany*, November 2005.

[14] A. Ledeczi, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason, G. Nordstrom, J. Sprinkle, and P. Volgyesi, "The generic modeling environment," *In Proceedings of WISP'2001, Budapest, Hungary*, pp. 255–277, May 2001.

[15] I. H. Krueger and R. Mathew, "Component synthesis from service specifications," *In Proceedings of the Scenarios: Models, Transformations and Tools International Workshop, Dagstuhl Castle, Germany, Lecture Notes in Computer Science, Vol. 3466*, pp. 255–277, September 2003.

[16] L. Lymberopoulos, E. Lupu, and M. Sloman, "An adaptive policy-based framework for network services management," *Journal of Network and Systems Management*, vol. 11 , Issue 3, pp. 277 – 303, 2003.

[17] A. Prakash, Z. Theisz, and R. Chaparadza, "Formal methods for modeling, refining and verifying autonomic components of computer networks," *Advances in Autonomic Computing: Formal Engineering Methods for Nature-Inspired Computing Systems, Springer Transactions on Computational Science (TCS)*, Expected Publication: Winter 2010 (accepted).

[18] G. Batori, Z. Theisz, and D. Asztalos, "Configuration aware distributed system design in erlang," *Erlang User Conference, Stockholm, Sweden*, 2006.

 ISBN: 978-1-61208-130-4

Towards Semantic Interoperability of Graphical Domain Specific Modeling Languages for Telecommunications Service Design

Vanea Chiprianov, Yvon Kermarrec
Institut Telecom, Telecom Bretagne
Université européenne de Bretagne
UMR CNRS 3192 Lab-STICC
Technopole Brest Iroise, CS 83818 29238
Brest Cedex 3, France
Vanea.Chiprianov@telecom-bretagne.eu

Siegfried Rouvrais
Institut Telecom, Telecom Bretagne
Université européenne de Bretagne
Technopole Brest Iroise, CS 83818 29238
Brest Cedex 3, France
Yvon.Kermarrec@telecom-bretagne.eu
Siegfried.Rouvrais@telecom-bretagne.eu

***Abstract*—High competition pressures Telecommunications service providers to reduce their concept-to-market time. To manage more easily service complexity among several actors in the design process and to ensure a more flexible maintainability, service decomposition into stakeholder dedicated views is now largely investigated by companies. However, there is still a lack of tools to fully support and implement this approach in various domains, especially Telecommunications. Consequently, in this position paper, we defend using a Domain Specific Modeling Language for each viewpoint. We also regroup them into a family of modeling languages, relying on a meta-modeling approach. To ensure better interaction and coherence between the various viewpoints, we focus on some interoperability issues early at design time. To adequately and systematically manage interoperability between distinct graphical models, interoperability between their meta-models should be established as well. For this we rely on model transformations between meta-models. However, most often model transformations address only the syntactic level. To increase the formality of languages and of their interoperability, semantics must be taken into consideration as well. Therefore, we propose lifting the meta-models into ontologies, enriching and matching them into shared ontologies. This allows for semi-automatic generation of model transformations from shared ontologies.**

***Keywords*-Interoperability, DSML, ontology, semantics.**

I. Telecommunications Service Design

Every time we call, send text or videos with smartphones, talk using a Skype©-like program or share documents using a secure connection, we are end-users of Telecommunications services (e.g., call, voice over IP, Virtual Private Network (VPN)). These services are delivered by service providers, more and more by operators. They use telecommunications, next generation or computer networks. Traditionally, before a service offers acceptable quality of service and can be launched to a market, it has to pass through several phases (e.g., from design, to implementation, test and deployment). These phases tend to be long and not sufficiently adapted to the current competitive market. More and more companies like Google© and Skype© appear on the service provider market, offering shorter time delivery for innovative services. Consequently, traditional providers are pressured to reduce their concept-to-market time for new services while still maintaining a high level of quality to guarantee a smooth integration with their infrastructure.

A. Viewpoints

To support the increasing complexity of new services and reduce their concept-to-market time, the International Telecommunications Union has introduced the Intelligent Network Conceptual Model (INCM) [1], as "a framework for the design and description of the Intelligent Network architecture". It consists of four "planes", or views, each refining the service definition from the upper-level plane. More recent proposals, like Enhanced Telecom Operations Map [2] for Telecom, or more general ones, like TOGAF [3] for enterprise architecture, also advocate reducing complexity through division into several layers or views. For greater designer usability, a Domain Specific Modeling Language (DSML) may be defined for each view.

A *Domain Specific Language* (DSL) is "a language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain" [4]. A *Modeling Language* is, "a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system" [5]. A *Domain Specific Modeling Language* (DSML) is therefore taken in this position paper to be a graphical language that offers, through appropriate notations and abstractions, expressive power focused on a particular problem domain, to visualize, specify, construct and document the artifacts of a software-intensive system.

A frequent approach to developing DSMLs is the Meta-Modeling approach [6], which defines a DSML as a set of:

- *Concrete syntax*: a human-centric representation of the syntax domain, which defines the symbols used to represent the concepts in the language;
- *Abstract syntax*: a computer-centric representation of the syntax domain;

 ISBN: 978-1-61208-130-4

- *Semantic domain*: the meaning of the language constructs;
- *Display mapping*: links the abstract to the concrete syntax;
- *Semantic mapping*: links abstract syntax to semantic domain.

The concrete and abstract syntaxes are usually defined as Meta-models (MMs). MMs play the same role for DSMLs as grammars for programming languages.

The display and semantic mappings can be defined as Model Transformations (MTs) [7]. A MT is the automatic generation of a target model from a source model, according to a set of transformation rules. A transformation rule is a description of how constructs in the source language can be transformed into constructs in the target language.

The semantic domain is the hardest to define. It may be defined through a semantic mapping towards the precise semantics of an existing programming language [7] so that tools can work on it. Dynamic semantics may be described through operational, denotational or axiomatic frameworks [8] and static semantics through ontologies.

B. Interoperability Issues

To ensure better interaction and coherence between various modeling viewpoints, we focus on interoperability (interop.) issues at design time. One DSML per design view favors in depth control of designers on a particular domain. However, having DSMLs for several views introduces interop. issues between the models designed, in an ideal top-down approach, with adjacent view DSMLs. Therefore, in what follows, we address the issue of ensuring semantic interop. between the models defined with two different DSMLs, which we instantiate to Telecommunications.

II. On Interoperability of Modeling Languages

There are numerous definitions for interop. in literature, depending on the domain. For our purposes, and following [9], we consider interoperability to be the ability of two or more *tools* to *exchange models* so as to use them in order to operate effectively together. Considering this definition, to operate together, tools for adjacent view DSMLs need to exchange models. Considering that models are conformant with MMs, and that, in a meta-modeling approach, MMs define (the syntax of) DSMLs (Sect. I-A), the issue of tools exchanging models written in different DSMLs becomes an interop. issue between DSMLs. So, to ensure interop. between models, one must address interop. between DSMLs.

Because interop. is a complex problem, there are numerous proposals of decomposing it into levels. One particularly suitable for our approach is the C4IF (Connection, Communication, Consolidation, Collaboration Interoperability Framework) [9]. This is due to its mapping between Information Systems (IS) Communication and Linguistics. Linguistics and the Meta-Modeling approach (Sect. I-A) share concepts (e.g., syntax, semantics), thus establishing the connection with C4IF. The C4IF defines four levels:

1) *Connection*: the ability of ISs to *exchange signals*.
2) *Communication* refers to the ability of ISs to *exchange data*. *Syntactic* communication includes data in commonly accepted data syntax/schemas.
3) *Consolidation* refers to the ability of ISs to *understand data*. The focus is on data meaning (i.e., *semantics*).
4) *Collaboration* refers to the ability of ISs to *act together*.

These levels of interop. are usually defined in such a manner so as to ensure a (strict) linearity [9] between them - to reach an upper level of interop., all the previous levels must have been successfully addressed.

In order to ensure interop. between two DSMLs we ideally have to ensure all four levels of the C4IF. The ISs of C4IF, in our case, are the tools associated to DSMLs, and the data they exchange, are thus the models. We consider the C4IF connection level as being implemented by existing communication and signaling media in computers.

The mapping proposed in [9] assigns the Communication interop. level of ISs Communication to Syntax of Linguistics. So, communication, *syntactic interop.*, between DSML tools, is the level of interop. between the syntaxes of DSMLs. Approaches to ensure syntactic interop. between different DSMLs have been proposed, like combining MMs [10]: extension, merge, embedding, weaving or hybrid approaches. However, we strongly recognize that the most flexible way to describe relations between two MMs, is through MTs. Using MTs, one can describe the similarity relations between two MMs and capture the intersection between the concepts of their respective DSMLs. Nevertheless, MMs describe only the syntaxes of DSMLs. So MTs, or other combination approaches between MMs, can describe interop. only at a syntactic level.

The mapping proposed in [9] assigns the Consolidation interop. level of ISs Communication to Semantics of Linguistics. So, consolidation, *semantic interop.*, between DSML tools, is the level of interop. between the semantics of DSMLs. We focus in what follows on semantic interop.. We do not yet address collaboration, as we consider, in conformance with their (strict) linearity property, that this level must be ensured first.

III. Towards Semantic Interoperability through Ontologies

Formal semantic description is significant for the design, reasoning and standardization of programming languages, ensuring their final unambiguous execution or interpretation. It is usually classified into static and dynamic. The frameworks for formal dynamic semantics are usually classified [8] as operational, denotational, or axiomatic. In surveying them, [8] concludes that ”compared to the amount of effort that has been made to the research of various semantic frameworks over more than forty years, their actual applications are definitely frustrating”. Therefore, even if there are approaches using formal semantics to address interop. in a family of DSLs [11], we do not tackle dynamic

 ISBN: 978-1-61208-130-4

semantics here. We restrict at static semantics and further investigate ontologies to describe it. Even if ontologies in a broader sense can also define "dynamic" concepts such as Process, State, Event, they are typically used to describe static concepts, and that is how we use them. We restrict here to using ontologies for static semantics and don't investigate using ontologies for dynamic semantics.

A. On the use of Ontologies with Meta-models

The common thread in defining ontology [12] is that it is a *formal description* of a domain, intended for *sharing* among different applications, and expressed in a language that can be used for reasoning.

To date, to the best of our knowledge, there is no common agreement on the relationship between MMs and ontologies in the scientific community. While many agree that MMs and ontologies share many and "deep" characteristics, there are also numerous highlighted differences, and some consider that MMs and ontologies are complementary [13]. Mostly, ontologies have been used with MMs for:

- *Model checking*: using automated reasoning techniques for validation of models in formalized languages.
- *Model enrichment*: expressing the semantics of modeling concepts whose syntax is defined by a MM.
- *Semi-automatic identification of mappings between MMs*: discovering mappings between MMs.

B. Ensuring Semantic Interoperability between Static Semantics of Modeling Languages

We propose to use ontologies for: describing the static semantics of DSMLs (i.e., model enrichment) and discovering a common reference ontology (i.e., semi-automatic identification of mappings between MMs). A common ontology will ensure semantic interop. and coherence between two adjacent view DSMLs. It can be discovered by determining the mapping between two ontologies, each describing the semantics of one DSML. For this, we promote this approach:

1) *Lift*. It transforms each MM into an ontology. We implement it through a MT between the meta-MM describing the modeling technical space (e.g., Ecore[1]) and the meta-MM describing the ontology space (e.g., OWL DL[2]). OWL DL is particularly suited for our approach, as its definition is already given in the form of a MM.

2) *Enrich*. The lifted MMs are enriched by applying patterns. Finding correspondences between relationships of different MMs can be addressed this way. Patterns similar to that of "Association Class Introduction" [14] can be used. A new class is introduced in the ontology similarly to an association class in UML, thus transforming relationships from MMs into concepts in ontologies. We implement it through an endogenous MT, with input and output the meta-MM describing the ontology space.

3) *Align*. In the ontology technical space we apply ontology-specific techniques [15] (e.g., alignment) on the lifted and enriched MMs of two adjacent views, thus discovering their intersection. Because the lifted and enriched MMs describe semantics of DSMLs, the discovered *shared ontologies* represent in fact the semantics of the MTs between the original MMs. Rediscovering these shared ontologies each time the (lifted and enriched) MMs describing static semantics of DSMLs evolve, is what we mean by ensuring (static) semantic interop. between two DSMLs.

4) *Generate*. MTs which have as input and/or as output other MTs are called Higher Order model Transformations (HOTs). We use shared ontologies as input for HOTs between the meta-MM describing the ontology technical space and the meta-MM describing the MT space (e.g., QVT[3]), which generate MTs between the original MMs.

Consequently, we can automatically generate and evolve MTs for a family of DSMLs, through their connections with shared ontologies, thus ensuring their syntactic and static semantic interop.. The whole process can be automatized and thus enables a high rate of reuse and faster iterations on evolving MMs.

C. Related Work

Kappel et al. [14] propose a process which semi-automatically lifts MMs into ontologies, refactors, enriches, and then applies ontology matching on them. However, unlike our approach, they do not use the discovered matchings to generate MTs. On a more technical point, they implement the lifting step by specifying a weaving model from which they generate ATL code, while we use MTs in QVT.

Hoss and Carver [16] propose connecting MMs with ontologies to assist in software evolution. While they connect MMs with generic ontologies, using what could be called an alignment strategy, we lift MMs into ontologies, using a generative strategy. Also, they have to create model weavings every time new (versions of) MMs are introduced. In our approach, MTs defined between meta-MMs (cf. e.g. Sect. IV) are sufficient for handling any MMs.

IV. TELECOMMUNICATIONS CASE STUDY

Figure 1 exemplifies the proposed approach on two MMs for the adjacent planes/views Global Functional Plane (GFP) and Distributed Functional Plane (DFP) of INCM (i.e., MM_{GFP} and MM_{DFP}). Each MM describes a DSML for VPN at GFP [17] and respectively DFP.

Each MM is *lifted* into an ontology (e.g., O_{GFP} and O_{DFP}) by means of a MT (i.e., $MT_{Ecore2OWLDL}$). This MT is sufficient for lifting any MM into an ontology, as it transforms concepts from Ecore, the language (meta-MM) in which MMs are written, into concepts from OWL DL, the language in which ontologies are written. To write this

[1]http://www.eclipse.org/modeling/emf, accessed 24th November 2010
[2]http://www.omg.org/spec/ODM/1.0/, accessed 24th November 2010
[3]http://www.omg.org/spec/QVT/1.0/, accessed 24th November 2010

 ISBN: 978-1-61208-130-4

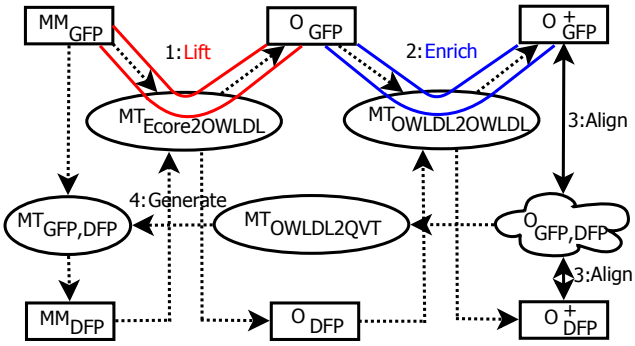

Figure 1. Syntactic and semantic interop. through MTs and ontologies.

MT, we build on the mapping provided by [14], updating it to the new versions of Ecore and OWL DL.

On each lifted MM (e.g., O_{GFP} and O_{DFP}), patterns for refactoring, checking and *enriching* are applied through a MT (i.e., $MT_{OWLDL2OWLDL}$). Similarly to lifting, this MT is sufficient for the enrichment of all lifted MMs.

The enriched ontologies (e.g., O^{+}_{GFP} and O^{+}_{DFP}) are *aligned*, resulting a shared ontology (e.g., $O_{GFP,DFP}$).

With the shared ontology as input, $MT_{OWLDL2QVT}$ *generates* the MT between the initial MMs (e.g., $MT_{GFP,DFP}$). Similarly to lifting and enrichment, this MT is sufficient for the generation of all MTs between the initial MMs.

Currently, we are writing MTs in QVT Relations. For ontology matching, evaluations [18] suggest ASMOV [19] as a good mature candidate tool.

V. Discussion

For Telecommunications, we defend that to manage interoperability between distinct graphical models in a viewpoint approach, interoperability between their meta-models should be established as well. For this we propose using model transformations between meta-models and lifting the meta-models into ontologies. As formulated in this paper, using a meta-modeling approach combined with ontologies has the advantage of co-evolving syntactic and semantic bridges that ensure interoperability between DSMLs. However, this co-evolution depends greatly on the shared ontology between views. If this would be poor or even empty, the interoperability bridge would be narrow. Consequently, in order for the proposed approach to be effective one should first make sure that the vocabularies for different viewpoints have a fair amount of concepts in common. This supports the idea that such an approach would be beneficial especially in the case of families of modeling languages.

References

[1] Study Group XVIII, *Principles of Intelligent Network Architecture. ITU-T Recommendation Q.1201*, International Telecommunication Union Std., 1992.

[2] TMF Forum, *Enhanced Telecom Operations Map (eTOM), GB921, Release 8.0*, TMF Forum Std., November 2008.

[3] The Open Group's Architecture Forum, *TOGAF Version 9 Enterprise Edition*, Std.

[4] A. Deursen, P. Klint, and J. Visser, "Domain-specific languages: an annotated bibliography," *SIGPLAN Not.*, vol. 35, no. 6, pp. 26–36, 2000.

[5] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language User Guide*. Reading, MA, USA: Addison-Wesley Professional, 2005.

[6] T. Clark, A. Evans, S. Kent, and P. Sammut, "The MMF approach to engineering object-oriented design languages," in *Wksh. on Language Descriptions, Tools and Applications (LDTA)*, 2001.

[7] I. Kurtev, J. Bezivin, F. Jouault, and P. Valduriez, "Model-based DSL frameworks," in *OOPSLA '06*, 2006, pp. 602–616.

[8] Y. Zhang and B. Xu, "A survey of semantic description frameworks for programming languages," *ACM SIGPLAN Notices*, vol. 39, no. 3, pp. 14–30, March 2004.

[9] V. Peristeras and K. Tarabanis, "The Connection, Communication, Consolidation, Collaboration Interoperability Framework (C4IF) For Information Systems Interoperability," *Intl. Jour. of Interoperability in Business Information Systems*, vol. 1, no. 1, pp. 61–72, 2006.

[10] A. Vallecillo, "On the combination of domain specific modeling languages," in *Proc. of the 6th European Conf. on Modelling Foundations and Applications (ECMFA)*, ser. LNCS, vol. 6138, Paris, France, 2010.

[11] I. Ober, A. Abou Dib, L. Féraud, and C. Percebois, "Towards Interoperability in Component Based Development with a Family of DSLs," in *Proc. of the 2nd Europ. conf. on Software Architecture (ECSA)*, Paphos, Cyprus, 2008, pp. 148–163.

[12] C. Welty, "Ontology research," *AI Magazine*, vol. 24, no. 3, pp. 11–12, 2003.

[13] M.-N. Terrasse, M. Savonnet, E. Leclercq, T. Grison, and G. Becker, "Do we need metamodels AND ontologies for engineering platforms?" in *Proc. of the Intl Wksh on Global integrated Model Management (GaMMa)*. New York, NY, USA: ACM, 2006, pp. 21–28.

[14] G. Kappel, E. Kapsammer, H. Kargl, G. Kramler, T. Reiter, W. Retschitzegger, and M. Wimmer, "Lifting metamodels to ontologies - a step to the semantic integration of modeling languages," in *Proc. of the ACM/IEEE 9th Intl Conf. on Model Driven Engineering Languages and Systems (MoDELS/UML)*, 2006, pp. 528–542.

[15] N. Choi, I. Song, and H. Han, "A survey on ontology mapping," *ACM Sigmod*, vol. 35, no. 3, p. 41, 2006.

[16] A. Hoss and D. Carver, "Towards Combining Ontologies and Model Weaving for the Evolution of Requirements Models," in *Innovations for Req. Analysis. From Stakeholders' Needs to Formal Designs*, ser. LNCS, 2008, vol. 5320, pp. 85–102.

[17] V. Chiprianov, Y. Kermarrec, and P. Alff, "A Model-Driven Approach for Telecommunications Network Services Definition," in *Proc. of the 15th Open European Summer School and IFIP TC6. 6 Wksh on The Internet of the Future*, ser. LNCS, 2009, pp. 199–207.

[18] J. Euzenat, A. Ferrara, C. Meilicke, J. Pane, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Svab-Zamazal, V. Svatek, and C. Trojahn dos Santos, "First results of the Ontology Alignment Evaluation Initiative 2010," in *Proc. of the 5th Intl. Wksh. on Ontology Matching (OM), with the 9th Intl. Semantic Web Conf. (ISWC)*, Shanghai, China, 2010.

[19] Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka, "Ontology matching with semantic verification," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 3, pp. 235 – 251, 2009.

 ISBN: 978-1-61208-130-4