A language for interoperability modeling and prediction

Johan Ullberg*, Pontus Johnson, Markus Buschle

KTH Royal Institute of Technology, School of Electrical Engineering, Department of Industrial Information and Control Systems, Stockholm, Sweden

ARTICLE INFO

Article history:
Received 15 November 2011
Received in revised form 22 July 2012
Accepted 1 August 2012
Available online 30 August 2012

Keywords:
Interoperability
Interoperability prediction
Information systems architecture
Interoperability modeling
Architecture analysis

ABSTRACT

Interoperability, defined as the satisfaction of a communication need between two or more actors, is a sought after quality for enterprises in today’s competitive environment. For a decision maker, understanding the effects of a changing market place and understanding how to adapt to the new environment is essential. Sustainable interoperability is an approach where such dynamic environments are considered, including how to adapt to the new environments. This paper presents a modeling language for describing architectures from an interoperability perspective and a formalism for inferring the degree of interoperability from the architecture models, thus supporting sustainable interoperability.

The interoperability language is expressed as a Unified Modeling Language, UML, class diagram specifying classes, attributes, and relationships relevant for interoperability modeling. The class diagram is also augmented with a set of statements in the Object Constraint Language, OCL, supporting automated interoperability prediction.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Interoperability is a sought after quality for enterprises in today’s competitive environment. Interoperability has been approached from many different points of view and perspectives [1], the concept of interoperability is however complex and fundamental interoperability problems are still not well understood [2]. In this paper, interoperability is defined as “the satisfaction of a communication need between two or more actors”, a definition compatible with the widely adopted definitions, e.g. IEEE’s, “the ability of two or more systems or components to exchange information and to use the information that has been exchanged” [3]. Sustainable interoperability is an approach to interoperability where the ever-changing environment of an organization is considered. The organization needs to learn about changes, analyze their impact on the current communication needs and investigate how to adapt the system and network in order to ensure satisfaction of current and new communication needs [4]. By allowing automated prediction, the work presented in this article can facilitate sustainable interoperability and thereby enable swift responses to changes in the communication needs.

Information system architecture is an approach to information systems management that relies on models of the information systems and their environment. Instead of building information systems using trial and error, a set of models is proposed to predict the behavior and effects of changes to the system. The architecture models allow reasoning about the consequences of various scenarios and thereby support decision-making. The chosen architecture models must contain relevant information for the issue at hand. Therefore, there is a need for a tailored modeling language of the various aspects of interest for the decision maker. Most current system architecture frameworks, however, lack modeling languages that support interoperability prediction [1].

This paper presents an architecture-based formalism for predicting the interoperability between information systems. The formalism is expressed in terms of a class diagram of the Unified Modeling Language, UML [5], in the remainder of the article referred to as a metamodel. This metamodel contains classes, attributes and relationships relevant for creating business and information system architecture models from an interoperability perspective. In order to enable interoperability prediction, the metamodel is coupled with a set of rules that need to be asserted in order to achieve interoperability, i.e. satisfying a communication need. This rule set is formally expressed in the Object Constraint Language, OCL [6].

To provide an example of the proposed rule set for interoperability prediction, consider a case of telephone communication between two people. A customer wants to order merchandise over telephone from a sales employee at a retail company. Fig. 1 illustrates this setup together with the languages spoken by the individuals. From an interoperability perspective, one could be interested in ensuring the following two rules. Firstly, is there

* Corresponding author. Tel.: +46 8 790 68 23.
E-mail addresses: johantu@ics.kth.se (J. Ullberg), pj101@ics.kth.se (P. Johnson), markusb@ics.kth.se (M. Buschle).
sufficient infrastructure for the information exchange? Secondly, will they understand the messages being passed between them, i.e. do they share a common language for information exchange? Based on the model, it is reasonable to believe that the infrastructure is in place in terms of their connection to the telephone grid. However, the communication will still fail since they do not share a language for communication. This model, through its underlying metamodel, thus contains relevant information for the task at hand, i.e. interoperability prediction. The interoperability metamodel presented in this paper is a more elaborate version of the one implicitly considered in this example.

The contribution of this paper is two-fold. Firstly, the interoperability metamodel describing the classes, attributes and relationships necessary to create architecture models from an interoperability perspective is outlined. Secondly, a rule set describing interoperability requirements is presented, thus enabling automated interoperability prediction on architecture models. The work has two main delimitations. Firstly, the prediction does not cover the dynamic aspects of interoperability, i.e. the order of the information exchange. Secondly, it focuses on enabling factors and does not cover preventive aspects. Preventive aspects are various mechanisms that actively block communication; mainly various security mechanisms e.g. access control.

1.1. Evaluation criteria for prediction frameworks

When developing tools and methods for prediction several quality criteria should be considered. Adapting the six requirements on decision calculus posed by [7] to prediction frameworks, the following four criteria can be stated: (1) Accuracy, i.e. that the models created based on the metamodel really are capable of predicting interoperability. (2) Cost of use, i.e. the effort needed to reach the goal, in this case interoperability prediction. This includes data collection, the creation of the architecture models and performing the predictions. (3) Cognitive complexity, i.e. the learning curve for applying the metamodel and rule set should be sufficiently low. Finally, (4) error control, is concerned with the trade-off between accuracy and cost of use that often is necessary, it is important to be able to make an informed decision regarding this trade-off. Using these criteria, it is possible to evaluate the metamodel and rule set presented in this article. Such evaluation will be performed in Section 5, below.

1.2. Outline

The next section discusses related works in the field of interoperability modeling and prediction. Section 3 describes the formalism that is used for interoperability prediction. The fourth section constitutes the main contribution of the paper. The section first presents a metamodel suitable for interoperability modeling and secondly the rule set needed for interoperability prediction. After this, Section 5 discusses the benefits of the presented formalism and finally, conclusions are drawn in Section 6.

2. Related works

The related works can be divided into two main categories, although not completely mutually exclusive. The first category is concerned with interoperability frameworks. Secondly, related work on assessing interoperability using maturity models or other approaches is considered.

Recently, several initiatives on interoperability have proposed interoperability frameworks targeted at structuring issues and concerns in quite different ways. The European Interoperability Framework in the eGovernment domain [8] defines three aspects of interoperability: semantic, technical and organizational. A similar approach was also used by the ATHENA Interoperability Framework (AIF), structuring interoperability issues and solutions at the three levels: conceptual, technical and applicable [9]. The Framework for Enterprise Interoperability [10] is another interoperability framework that focuses on barriers to interoperability. All these interoperability frameworks provide means to classify the interoperability problems and solutions. At the same time they lack the ability to describe the interoperability situations where the problems occur and are solved.

The ontology of interoperability (Ool) [11] prescribes a set of metamodels to describe interoperability from various viewpoints, mainly aiming at classifying various problems and decision alternatives. Ool does however also provide a communication metamodel, aimed at describing architectures from an interoperability perspective. The metamodel described in this article uses several of the concepts of Ool. The work presented in this article also provides a means to predict interoperability, something not offered by the Ool.

Several methods for assessing interoperability have previously been suggested. The Levels of Information Systems Interoperability (LISI) [12] uses a maturity model for assessing interoperability. The assessment in LISI is based on an assessment process and utilizes a scorecard method and interoperability metrics. Employing LISI would require more domain knowledge in the field of interoperability than the method presented in this paper. The same is true for several other similar approaches, such as Systems of Systems Interoperability (SoSI) [13] and Levels of Conceptual Interoperability Model (LCIM) [14].
The i-Score [15] is a methodology for quantitative interoperability assessment. The assessment is based on the concept of operational threads; a sequence of activities each supported by exactly one system. Such threads are used in order to calculate an i-Score. Compared to the metamodel and assessment methods in this paper, i-Score requires deeper understanding of the interoperability field and lacks the describing capabilities of the metamodel in this article.

Summarizing, there is much work done in classifying interoperability issues in various frameworks. There are also some approaches to measuring interoperability. However, these all require more knowledge in the field of interoperability than the approach in this paper, where the prediction is formalized in terms of OCL statements and thus can be automated, something of great value for sustainable interoperability where rapid response to changes in the environment is necessary [4]. Furthermore, none of the available approaches allow for a trade-off between prediction accuracy and cost through a probabilistic approach as the work in this paper.

3. A formalism for architecture prediction

The dominating notation for software and systems modeling today is the Unified Modeling Language (UML) [5]. Practically all major software architecture and design tools, as well as many system architecture tools, are based on or support UML modeling [16]. UML specifies several different model notations of which the class diagram is one of the most well-known [5].

The Object Constraint Language (OCL) is a formal language used to describe constraints on UML models. OCL expressions typically specify invariant conditions that must hold for the system being modeled, or queries over objects described in a model [6]. Such queries can be seen as gathering structural information from the model. Referring to the introductory example of Section 1, a metamodel where actors and the languages these actors understand are modeled as separate entities, one relevant query could be "given two actors with a need to communicate, do these actors have a common language?". Such queries can be expressed in OCL. More precisely, OCL is capable of expressing prediction theories based on first-order logic, arithmetics and set theory [6].

UML and OCL together constitute the formalism used to express the metamodel for interoperability prediction presented in this article. In order to enable modeling from an interoperability perspective, a UML class diagram has been developed, i.e. level M1 in the MOF hierarchy, referred to as a metamodel. This metamodel contains classes, relationships and attributes relevant for modeling interoperability related aspects on the M0 level. The metamodel is augmented with OCL statements. These statements describe rules on the M1 level that need to be asserted in order to achieve interoperability, e.g. that two actors must share a common language. These rules are then applied on models on the M0 level to predict the interoperability of that particular scenario.

OCL, however, is a deterministic language, and as information systems are rapidly growing more complex, i.e. growing in numbers, size and in the complexity of the underlying technologies, a deterministic approach becomes difficult to apply. While it a few decades ago was feasible for one person to fully grasp the workings of any information system, this is no longer the case simply due to the increased complexity. Furthermore, the poor state of documentation that plagues many projects adds to our uncertainty. Thirdly, the use of externally operated information systems, e.g. cloud services and applications, is increasing. Finally and perhaps unrelated to the IT industry’s maturation, in the early development phases, many aspects of the information system to be developed are uncertain. For these reasons, it would be of great benefit for the modeler to be able to express his confidence in the created model, and for this confidence to reflect in the result of the prediction. [17]

Although the rule set described in this article can be used for deterministic interoperability prediction, it is shown in [17] that standard OCL statements can be evaluated probabilistically. This however, poses two additional requirements on the architecture models. Firstly, the attributes of the metamodel should be specified as random variables. The modeler should be able to specify the value of e.g. the attribute availability of a network, not as true or false, but rather as a probability distribution. Secondly, it should be possible to express structural uncertainty, i.e. uncertainty regarding the existence of objects and relations. For this purpose, all classes and relationships (and thereby also all objects and relations) must feature an additional Boolean existence attribute, indicating the modeler’s belief in the structure of the model. The modeler might for instance be uncertain of whether actor A really communicates with actor B, or whether actor B really can use the format X. Probabilistic evaluation of OCL statements for architecture prediction is further described in [17] and Section 5 evaluates OCL, including the probabilistic evaluation, with respect to the requirements on prediction formalisms as presented in Section 1.1.

4. A metamodel for interoperability prediction

In this section, a metamodel for interoperability prediction is presented. The metamodel is divided into two main parts, the structural aspects for interoperability and the conversation-specific aspects, represented as white and shaded classes of Fig. 3 respectively.

- Structural aspects cover the basic infrastructure for interoperability. They detail, for instance, the parties that are to interoperate, the format with which the information is encoded and similar aspects.
- The conversation-specific aspects are a more fine-grained description of a particular conversation detailing the messages being sent between parties, the content of such conversation, etc.

The classes related to the structural aspects can be used autonomously whereas the conversation-specific classes are a refinement requiring the structural aspects as a fundament. The conversation-specific classes allow for a more in-depth description and interoperability prediction.

This section is the main contribution of the paper and is outlined as follows: firstly, a brief overview of the metamodel is provided for orientation purposes. After that, the metamodel is described in more detail, presenting each class, relationship and attribute. This is followed by a subsection that illustrates how the metamodel can be used for models on different levels of granularity. In the final two subsections, the focus is shifted to interoperability prediction, describing the rule set necessary for this prediction as well as the additional rules needed to allow modeling on various levels of granularity.

4.1. Overview of the metamodel

In order to derive the metamodel presented in this section, a literature review was performed. From the literature, rules for ensuring interoperability were extracted and translated into the rule set. Based on this rule set, the concepts relevant to include in the metamodel were elicited. The metamodel was thus created in the opposite order to how it is presented, and the rationale for the modeling concepts presented in Sections 4.1–4.3 can therefore be found in the description of the rule set. Section 4.5. In the remainder of the article, boldface font will be used to refer to concepts of the metamodel.

The definition of interoperability used in the proposed metamodel can be expressed as “the satisfaction of a communication
need between two or more actors’. The authors believe that this definition is compatible with that of the IEEE, “the ability of two or more systems or components to exchange information and to use the information that has been exchanged” [3], albeit with a somewhat wider definition of “systems and components”. Actors can take various forms such as systems, components, humans and whole enterprises but they all share the ability to actively use information, i.e. to operate on it, interpreting it, transforming it, etc. Communication Need satisfaction requires information exchange, which in turn necessitates a medium for transmitting the information. Examples of such media, or Message Passing Systems, are the Internet or Ethernet in computer communication or air in spoken communication between Actors of close distance. Translating this to classes of a metamodel, see Fig. 2, the three concepts mentioned above correspond to the classes Communication Need, Actor and Message Passing System respectively.

Actors identify other Actors using an Address, e.g. the name of a person. Furthermore, the Actors encode the information in a format, or Language. Examples of such Languages could be the IEC Common Information Model for information exchange between electric utilities, or spoken English when using the air as medium. Actors can also translate between different Languages, i.e. Language Translations. One example is a message broker in an integration platform. Furthermore, Message Passing systems use Languages for transporting information (i.e. the protocol), such as SOAP in a service oriented environment.

A special Language is the Reference Language, a Language in which all the concepts relevant to the Communication Need can be unambiguously defined. Considering Fig. 2, only the class Abstract Actor remains to be explained. Abstract Actor is an abstraction of both Actor and Message Passing System describing the common attributes and relationships of these classes.

The remainder of this section will describe the metamodel in more detail, including the additional concepts needed for modeling conversation-specific aspects, and also describe the rule set for interoperability prediction.

4.2. Structural aspects

Recall that an interoperability prediction corresponds to evaluating an instance of the class Communication Need. This class relates a set consisting of at least two Actors who share a desire to exchange information on a certain topic, to each other. A Communication Need is formulated in a special Language, the Reference Language. The Reference Language is a language in which the Communication Need can be expressed and, more importantly, evaluated. It is not necessarily the Language used in the communication. For more on Reference Languages, see Section 4.2.2 below. The attribute satisfied of class Communication Need is the target of the prediction, i.e. whether interoperability is achieved, and its value is derived using the rule set described later in this section.

The rest of the classes in the interoperability metamodel will be described in four main groups, starting with the classes that ensure a communication path between the Actors sharing a Communication Need.

4.2.1. Communication path

Two classes are required to establish a communication path, Actors and Message Passing Systems. In this subsection, we detail these classes. The class Actor describes a physical or virtual entity that actively participates in an interaction. Actors encode messages according to one or more Languages and can perform Language Translations. Actors cannot be related directly to each other, there needs to be a transport medium, denoted Message Passing System (MPS), between them.

In comparison to Actors, MPSs are passive elements that can be considered as channels transmitting messages in certain

---

Fig. 2. Simplified metamodel for the structural aspects of interoperability showing concepts necessary for all applications.
Languages from speaker to listener. It is not possible to couple two or more MPSs directly to each other. There needs to be an active instance (an Actor) that takes the output of one MPS and utilizes this as the input for the second MPS. An MPS is associated to one or several Languages, indicating which formats are allowed for message passing, i.e. the protocols according to which messages are transported by the MPS.

MPSs also have a second relationship to the Language class, addressing Language, expressing that the addressing of a certain MPS needs to be performed according to a special Language. Furthermore, MPSs can be separated in two categories, those where addressing is necessary and those that does not require addressing. The latter category contains MPSs where all Actors connected to the MPS are the intended parties of the communication, as in a private dinner conversation. Which category is relevant for a given MPS is indicated by the attribute fixed.

Actors and MPSs are specializations of the class Abstract Actor. The prefix abstract indicates that this class cannot be instantiated but rather contains the common properties of its specializations. Abstract Actors define the relationship to Language already described for the specializations. The Abstract Actor has three quality attributes that affect the information exchange. The Abstract Actor might lose information, modify data so that it becomes unusable or otherwise be hindered to take part in the information exchange. These three qualities are reflected in the three attributes dropMessage, distortMessage, and isAvailable.

4.2.2. Language

Languages are used to encode and transmit information. A Language might consist of several sub-languages. A sub-language can be fully expressed by the corresponding super-language, e.g. HTML as a subset of XML with a set of specific tags. A Language can also be a carrier of other Languages. This can be compared to a protocol for transmitting Languages, e.g. TCP being able to transmit XML (and thereby also HTML).

A Language can be mapped to other Languages, by the use of Language Translations. This procedure needs to be performed by a translator, i.e. an Actor. As Language Translations might be performed incorrectly, the attribute correct of the Language Translation describes the quality of the translation. Language Translations are necessary whenever two Actors share a Communication Need, but lack a common Language to communicate it in.

When considering the semantic correctness and completeness of a Language, we must compare the Language’s (artificial) Constructs to the (real) things we wish to express. In order to represent those things, the class Reference Language is introduced. The special characteristic of this language is that it must be able to express the involved Actors’ Universe of Discourse, i.e. the complete range of objects, events, attributes, relations, ideas, etc., that are intended by any Actor in a communication. Normally, the Reference Language is not used for actual communication, but simply as a tool to unambiguously evaluate the outcome of a communication; whether a Communication Need really is satisfied is thus evaluated in this Reference Language. As an example, reality is one possible Reference Language that may be used if the Communication Need is concerned with altering the reality, e.g. an enterprise receiving physical items from a supplier.

4.2.3. Addressing

Actors also need to be identified, which is achieved using Addresses, such as an IP address on the Internet. The Actor class has two relationships to the Address class: identifier and knownAddress. The former associates an Actor with an Address for identification whereas the latter constitutes the set of such identifiers known by a specific Actor.

If direct knowledge of Addresses is lacking, this barrier can be mitigated by the use of an address broker, e.g. a UDDI. Such an address broker is realized by one or several Actors, which are linked through MPSs. Such situations can be expressed as a separate Communication Need between the Actor that is lacking information and the address broker.

4.3. Conversation-specific aspects

In addition to modeling the structural aspects described above, it is also possible to model conversation-specific aspects. Conversation-specific models detail a particular message exchange between Actors. For this purpose the classes Conversation, Conversation Communication Need (CCN), Construct and Conversation Translation of Fig. 3 can be used.

A Language can be described in detail by its containing Constructs. Each possible message that could be formulated in a Language is represented as a Construct. During a message exchange, Actors exchange several Constructs. This collection of transferred Constructs is called Conversation. A Conversation might be dropped, which means the transmission of the Conversation is terminated and the message is no longer consumable. It might also be distorted, reflecting that it has been unintentionally modified so that the original meaning is lost. These two characteristics are reflected in the properties isDropped and isDistorted.

MPSs carry Conversations from one Actor to one or several others. To identify the target of a Conversation, the Conversation is coupled with Addresses, designating the receiving Actors of a Conversation.

A CCN describes the intention of a set of Actors to engage in a Conversation. This class is the conversation-specific equivalent of the class Communication Need. The class has the same attribute, satisfied, as the Communication Need, indicating a successful information exchange.

Finally, the class Construct Translation represents a mapping and transformation of a Construct of one Language into one of another Language. This mapping needs to be performed by an Actor. A Language Translation can be broken down into several Construct Translation instances.

4.4. Model abstraction

As stated in the introduction, the trade-off between prediction accuracy and modeling cost is an important aspect. The metamodel presented in this article features two types of abstractions in order to support such trade-off. Firstly, there is a choice between either modeling only the structural aspects, or modeling both the structural and the conversation-specific aspects. The latter, more comprehensive, modeling enables higher accuracy in the predictions. Secondly, the aggregation relationship of the Abstract Actor enables the models to be described using various levels of granularity. As previously described, Actors can take various forms, such as whole enterprises, departments or information systems. A more coarse-grained Actor, such as an enterprise, is generally composed of several more fine-grained Actors such as information systems.

4.5. Rule set for interoperability prediction

With the metamodel described in the previous subsection, it is possible to describe specific scenarios of communication between Actors. Given such a scenario, i.e. an instantiated metamodel, it would be possible for an interoperability expert to determine if the requirements for interoperability are fulfilled, and point out where the potential problems exist. The use of such experts is, however,
expensive, and automating the work performed by the expert is desirable. Formalizing theory for interoperability prediction using OCL statements, enables automatic assessment of the rules for ensuring interoperability.

This subsection describes the set of rules, in terms of OCL expressions, needed to predict interoperability, i.e. the satisfaction of a Communication Need, with references to literature justifying these rules. The expressions are described in natural language with references to [18], where the actual OCL code can be found. The interoperability requirements are grouped in the same way as the description of the metamodel, i.e. (1) communication path requirements, (2) language requirements, (3) addressing requirements and (4) requirements for specific conversations. The value for the attribute satisfied of class Communication Need, i.e. the target of the prediction, is evaluated by a pair-wise prediction of the Actors involved in the Communication Need, see statement 1 in [18]. For each such pair, queries to the model are performed to evaluate if information exchange is possible.

4.5.1. Communication path

The perhaps most fundamental requirement for interoperation to take place is that there is a path between the Actors that are to collaborate. For computer-based communication, a good example is the OSI reference model [19]. From the point of view of one OSI layer, the next lower layer of the OSI stack typically constitutes such a path, and the physical layer is the most basic path between Actors. Expressed in terms of the metamodel, this means that between the Actors associated to a Communication Need there must be a path of Abstract Actors. The possible paths between each pair of Actors are found and evaluated using statement 2 in [18], this statement starts with one of the Actors of the Communication Need and traverses the architecture model to find the target Actor. Every time a new Actor is reached, the allowed message formats (i.e. Languages) and a set of necessary conditions are, as described below, evaluated for this step in the path. Statements 8 and 3 capture, for each such step of the path traversal, the evaluation of allowed Languages and the additional necessary conditions
respectively. If there is no path to the target Actor statement 2 returns false (resulting in the same value for the satisfaction of the Communication Need).

Even if there is a communication path as described above, communication could still fail due to various barriers. First of all, the path may be unavailable [20,24]. For this reason, Actors and MPSs feature the attribute isAvailable. For each new Actor reached in the search for a path, this attribute will be evaluated for the new Actor and the intermediary MPS, see statement 4 in [18]. In the same manner, it is also possible to evaluate if any of the Actors or MPSs syntactically distorts or drops messages transmitted over the path [21,22]. Such evaluation is based on the attributes distortsMessage and dropsMessage respectively, cf. statements 5 and 6 respectively.

4.5.2. Languages
As has been described earlier, both Actors and MPSs use Languages. The Actors that are to communicate either need to share a Language for the messages passed between them or use a translator [20–22]. To capture these aspects, a list of the allowed Languages is kept while traversing the model. An allowed Language is a Language that can be used for message exchange between two Actors. The list of allowed Languages changes while traversing the model and can decrease, e.g. if an Actor along the communication path is not capable of using an allowed Language, and increase by the use of a Language Translations, see statement 8 where the list of allowed Languages is updated while traversing the model.

If Actors share a Language, the Language of the intermediary MPS also needs to be compatible with, i.e. a carrier of this Language [20,24]. Furthermore, the Language needs to be sufficiently expressive with respect to the information that is to be exchanged. These requirements are listed in statements 11 and 12 in [18] and are combined in statement 9 where only Languages that fulfill all the requirements are kept in the list.

Language Translations may be used to increase the set of allowed Languages [20,23]. In order to determine whether such a Language Translation is present, statement 10 in [18] is used. This statement finds all available Language Translations and for each evaluates the following criteria: the first requirement is that the Language Translation translates between Languages relevant to, i.e. used by, the collaborating Actors. Furthermore, one of the Languages in a translation needs to be an allowed Language and the translation also needs to be correct. Finally, these Languages need to be compatible with the MPS, and be able to express the information to be exchanged, i.e. express the concepts of the Reference Language.

4.5.3. Addressing
A third important requirement for interoperability is the addressing of parties involved in a collaboration. The most basic case is when the Actors of a Communication Need use an MPS that does not require addressing [19,24], e.g. a cable using between two information systems using serial communication, captured by the fixed attribute of the class MPS, cf. statement 15.

If the situation requires addressing, one possibility is that each Actor knows the Addresses of the other Actors, and the Addresses are expressed in a valid addressing language for the MPS [19,20]. This is modeled using the knownAddress relationship, and the two conditions are checked using statements 16 and 17 respectively.

If a new Actor (the source actor) knows the Address of another Actor (the target actor), but this Address is expressed in a Language that cannot be used on the MPS, there is also the requirement that there must be a third Actor (the translator) that can translate the known Address to a valid Address for the target Actor [20,24], see statement 18. The Domain Name Service (DNS), which translates a host name to an IP address [25], is an example of this situation.

The statements above are combined in statement 14 capturing static addressing, i.e. when addresses are explicitly, or (as in the DNS case) implicitly, known by the Actors. To capture dynamic addressing, e.g. when using an UDDI in a service oriented architecture [26], it is necessary to introduce a subordinate Communication Need for the addressing issue. This Communication Need is related to the main Communication Need through the addressingNeed slot. Such a sub-Communication Need is evaluated using the same statements as described in this section.

4.5.4. Conversation-specific aspects
When considering a specific conversation, a more detailed prediction can be performed utilizing the additional concepts of Conversations, Constructs, etc. as detailed above. The goal is, as with the structural aspects, to predict the value of the attribute satisfied, this time however, for the class Conversation Communication Need, CCN. Statement 19 assigns the value to this attribute.

4.5.4.1. Path. On the conversation-specific level, it is still necessary to ensure the existence of a path between the Actors of a CCN, see statement 20. As with the structural aspects, the other requirements (cf. statement 21) affecting the satisfaction of the CCN, are evaluated for each new Actor reached during the discovery of a communication path, see statement 20. The first requirement is that it is necessary to ensure that the path between the Actors is capable of transporting the Conversation [19,24]. Statements 29 and 30 express this, where it is ensured that each of the elements in the path between Actors is a part of the Conversation.

4.5.4.2. Language and Constructs. A Conversation consists of a set of Constructs. A basic requirement of these Constructs is that they can be transmitted on the MPSs [20,24]. Statement 22 ensures that the Language of the MPS is a carrier of the Language of the Constructs in the Conversation and thus can carry the Conversation. Construct Translations can be used to map concepts of two Languages. It is however necessary to ensure that these translations are properly carried out. This is evaluated in statement 28 by evaluating the attribute correct of the involved Construct Translations.

When it comes to the conversation-specific level, a Conversation can be dropped and distorted. Such distortion and dropping is the result of the actions of the Actors that are involved in the Conversation, and the MPSs that these Actors use to communicate the Conversation. In particular the attributes dropsMessage and distortsMessage of these classes are the ones that influence the corresponding qualities of the Conversation. Since these attributes are concerned with failure, it is sufficient that any of the Actors or MPSs fail for the Conversation to fail, see statements 25 and 27 respectively. This aggregated property for the Conversations in turn affects the successfulness of the CCN, see statements 24 and 26 respectively.

4.5.4.3. Addressing. With respect to addressing, it is also of importance that the target Address of a Conversation is an Address of one of the Actors involved in the communication [20,24]. This is expressed using statement 23 in [18].

4.6. Attribute aggregation for model abstration

When Actors and MPSs are combined into composite Actors and MPSs, the attributes distortsMessage and dropsMessage and isAvailable of the aggregate instance become dependent on the more detailed instances. These dependencies, found in statement 29, are based on the aggregation relationship of Abstract Actors. The attributes distortsMessage and DropsMessage utilize an OR
aggregation function, illustrating that it is sufficient that one of the composing Actors or MPSs distorts or drops the message respectively. For the isAvailable attribute, an AND aggregation function is used, i.e. all the sub components must be available in order for the aggregate Actor or MPS to be available.

This concludes the description of the metamodel and rule set, for an example application, see [28]. Employing the metamodel for prediction without additional support would be difficult. In particular the evaluation of the OCL statements would be cumbersome. To aid in this, a tool for enterprise architecture modeling and prediction has been developed [27] that is capable of such predictions.

5. Discussion

This section will evaluate the work presented in this article using the four quality criteria for prediction frameworks as described in Section 1.1, accuracy, cost, cognitive complexity and error control. The metamodel presented in this article meets these criteria by featuring the following seven properties: (1) formalized prediction theory; (2) automated prediction; (3) metamodel and rule set tailored particularly for interoperability prediction; (4) expressiveness of the formalism; (5) reusable prediction theory; (6) probabilistic predictions and finally (7) abstraction mechanisms of the metamodel.

The first property targets the criterion relating to prediction accuracy. Models created using the metamodel have a predefined means for interoperability prediction. Since the rules for prediction are formalized in OCL, the prediction will not feature errors due to e.g. misinterpretation, and prediction accuracy can thus be improved. Secondly, formalized rules allow for an automated prediction using a software tool [27]. This reduces the cost for prediction and improves the accuracy by eliminating human errors in the prediction. From an evolutionary perspective, it should be noted that since the prediction is based on architecture models, as the system of systems evolve, the architecture models need to be updated, with an associated cost. The actual discovery of evolution and maintenance of the many evolutionary stages (e.g. several potential to-be scenarios) is an important aspect, which will at minimum require software support, but is beyond the scope of this work. The mere use of architecture models is however one general approach to managing such evolution [16].

Furthermore, the metamodel presented in this article is based on class diagrams of UML [5], extended with concepts relevant for interoperability. This facilitates a more efficient modeling, thereby reducing cost of use. This specialization of the metamodel is based on state of the art research in the interoperability domain. This in turn ensures that all aspects relevant to interoperability prediction have been modeled and thus increases the prediction accuracy. The compatibility with the more generic modeling language of UML is also beneficial in order to reduce the cognitive complexity.

Fourthly, to enable prediction accuracy it is also important that the chosen formalism is sufficiently expressive for the issue at hand. Many interoperability concerns are related to conformance checking, e.g. that two communicating parties have a common communication format. From a business and information system architecture perspective, this poses requirements on the ability to describe the structure of the architecture, i.e. perform queries for structural information of the created models. The use of OCL for formalizing the rule set enables such structural queries as well as logic and arithmetic operations and is thereby sufficiently expressive for interoperability prediction.

Furthermore, the rule set for interoperability prediction is coupled with the class level concepts, i.e. with the metamodel. This enables the reuse of prediction theory in several application instances, i.e. instance models. Thus there is a separation of concerns between the prediction theory and the application of that theory. Enabling reuse of prediction theory reduces the need for costly domain experts. Allowing the end user to be agnostic regarding the rule set for prediction also reduces the cognitive complexity and allows the user to focus on the modeling aspect. The current rule set is targeted at static interoperability prediction. However, some investigation has been performed with respect to extending this to prediction of dynamic interoperability as defined in the introduction. Although this is work in progress, current findings suggest that the proposed set of classes are sufficient, there is a need for a new relationships between these classes to express temporal order and the rule set need to be expanded to fit such predictions.

Additionally, as described in Section 3, and more elaborately in [17], the rule set used for interoperability prediction can be evaluated probabilistically. Attribute values used in this prediction can thus be set coarsely using little data collection, or more precisely using additional resources. Furthermore, the modeler is allowed to express uncertainty regarding the structure of the architecture. The main benefit of a probabilistic approach is that it enables error control, i.e. a trade-off between the cost of use and prediction accuracy but also that it allows to manage situations where completely accurate data collection unfeasible and to indicate this uncertainty in the predictions, i.e. the trade-off is made explicit to the user.

Finally, the metamodel is capable of two abstraction mechanisms. The first such abstraction mechanism allows the modeler to describe interoperability on two levels, either on an abstract level where the structural aspects are covered – i.e. the structural and conversation-specific aspects. The latter allows for better predictive accuracy but also consume more resources in terms of modeling effort. Furthermore, the structural aspects can be modeled on different levels of granularity. For instance, an actor, such as an enterprise, participating in an information exchange, can either be described as one aggregate entity or refined as various sub-actors. Once again, this enables the modeler to perform a trade-off between the predictive accuracy and the additional cost necessary to both collect the more detailed data and perform modeling of it.

6. Conclusions

This article has demonstrated how an interoperability metamodel and a rule set for interoperability theory can be used as a foundation for achieving sustainable interoperability by enabling automated interoperability predictions. Although such prediction can be performed deterministically, the rule set presented in this paper can also be evaluated probabilistically, allowing the user to express uncertainty regarding the modeled concepts and perform a trade-off between prediction accuracy and modeling cost. From the perspective of sustainable interoperability [4] automation as well as the trade-off is important since it is essential to swiftly respond to changes in the environment, often with sparse knowledge.

Employing the work described in this article would, particularly aid in the learning capabilities of sustainable interoperability. Once a change to the environment is detected, it is possible to learn which part of the rule set that is no longer fulfilled and identify the areas in which adaptations need to be performed in order to restore interoperability. From an enterprise perspective, this would aid the a decision maker in the understanding of the as-is scenario and perhaps more importantly understand the transient effects from a change in the environment and how to, from an interoperability perspective, adapt to such new future scenarios.

The metamodel was developed to be generic and capable of describing many different interoperability scenarios. By specializing the metamodel as described in [29], e.g. specializing the class
MPS to the Internet, a general LAN or a specific LAN setup, a more detailed prediction can be performed. This is the case since the attributes of the specialized classes and the dependencies among them can be set with greater accuracy thus enabling a more precise prediction. The metamodel presented in this article is delimited to prediction of static interoperability, although, as described above, extensions to dynamic interoperability seem feasible. The planned approach to achieve this is to start eliciting the new requirements necessary, e.g. from current research in the field, and then mapping them onto the concepts of the modeling language to identify the new rules necessary for prediction.

References


Johan Ullberg received his M.Sc. in computer science at the Royal Institute of Technology (KTH), Stockholm in 2007. He is currently a Ph.D. student at the Department of Industrial Information and Control Systems at KTH. His research is focused on interoperability in technical support systems and in particular interoperability prediction based on architecture models.

Pontus Johnson is Professor and Head of the Department of Industrial Information and Control Systems at the Royal Institute of Technology (KTH) in Stockholm, Sweden. He received his M.Sc. from the Lund Institute of Technology in 1997 and his Ph.D. and Docent titles from the Royal Institute of Technology in 2002 and 2007. He was appointed professor in 2009. He has chaired and co-chaired a number of international conferences and workshops and participated in the program committees of approximately fifty such events. He has been associate and guest editor to several journals. He is secretary of the IFIP Working Group 5.8 on Enterprise Interoperability.

Pontus holds undergraduate courses and supervises Ph.D. students (currently six) at the Royal Institute of Technology. He has authored close to 100 scientific articles, mainly on the prediction of non-functional properties in software-intensive system architectures.

Markus Buschle received his M.Sc. degree in computer science at TUB, Berlin Institute of Technology, Germany. He is currently a Ph.D. student at the Department of Industrial Information and Control Systems at KTH – the Royal Institute of Technology Stockholm Sweden. His research focuses on the development of languages for enterprise architecture analysis and how they could be supported tool based.