A Modeling Language for Interoperability Assessments

Johan Ullberg, Pontus Johnson, Markus Buschle

A Modeling Language for Interoperability Assessments

Johan Ullberg, Pontus Johnson, Markus Buschle

Industrial Information and Control Systems, KTH Royal Institute of Technology,
Osquldas v. 12, SE-10044 Stockholm, Sweden
{johanu, pj101, markusb}@ics.kth.se

Abstract. Decision-making on issues related to interoperability can be furthered by the use of models of the organization or information system where interoperability is of concern. In order to provide decision-making support, the models should be amenable to analyses. This paper presents a modeling language specifically for interoperability issues where interoperability is defined as the probability that two more actors will be able to exchange information and use that information. The language is coupled with a probabilistic mechanism for automated interoperability assessments of the models created. The paper also presents an example of how the language can be applied.

Keywords: Interoperability, Modeling Language, Interoperability Assessment

1 Introduction

Interoperability is a sought after quality for enterprises in today's competitive environment that has been approached from many different points of view and perspectives [1]. Several definitions of interoperability have been proposed, one of the most well known and the one employed in this article is that of IEEE, "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" [2]. Based on this definition interoperability can be seen from the perspective of a decision maker as the problem of ensuring the satisfaction of a set of communication needs throughout the organization.

Enterprise architecture is an approach to enterprise information systems management that relies on models of the information systems and their environment. Instead of building the enterprise information system using trial and error, a set of models is proposed to predict the behavior and effects of changes to the system. The chosen architecture models must contain relevant information for the issue at hand. In the case of interoperability one important aspect is the information models, how messages are semantically and syntactically encoded. An architecture model describing how information is encoded, i.e. by containing relevant entities such as information models or protocols, is better suited for interoperability purposes than one lacking such information. Therefore there is a need of a tailored modeling language for representing the various aspects of interest for the decision maker. Most current enterprise architecture proposals, i.e. enterprise architecture frameworks, however

lack such modeling languages that allow reasoning. In particular languages for describing architectures from an interoperability perspective are not available. [3]

Furthermore, the decision maker generally needs to decide on future (to-be) architectures and in order to facilitate this process there is a need for methods and tools that support the evaluation of interoperability in the enterprise. Such methods and tools are sparse in the field of enterprise architecture as well as in the field of interoperability [3]. Currently such analysis would generally have to be performed by a domain expert, a costly approach to analysis. Automating the analysis of architecture models would thus be of great benefit to the decision maker.

The contribution of this paper is twofold; firstly it defines a modeling language for describing architectures from an interoperability perspective. Secondly the modeling language is coupled with an assessment mechanism for interoperability, allowing the user, e.g. the decision maker of a large organization, to perform analysis of the created models without extensive knowledge of the interoperability domain. The modeling language is expressed in terms of a probabilistic relational model (PRM) [4] that, apart from constituting the modeling language, also specifies an analysis mechanism for the created models. This mechanism is however insufficient to express all types of interoperability concerns and is in this article augmented with statements written in the Probabilistic Object Constraint Language, P-OCL [5], see Fig. 1 below for an overview of how these concepts relate to each other.

2 Related Works

The related works can be divided into three main categories, although not completely mutually exclusive. The first category pertains to modeling in general and the second is concerned with interoperability frameworks. Finally work on assessing interoperability using maturity models or other approaches are of relevance.

There exists many architecture modeling frameworks and languages. The foremost software system modeling language is UML [6]. The language provides a very generic metamodel that can be used for system design and analysis. Apart from the basic language there also exists several extensions to UML, such as SysML. With such a general language as UML there is no, or only little, guidance in what to model and there is little support when it comes to interoperability analysis.

Furthermore a substantial number of enterprise architecture frameworks are available that, apart from the information system domain, also take business and the usage of systems into account. Examples are the Zachman framework [7], the Department of Defense Architecture Framework (DoDAF) [8] and Archimate [9]. These languages all provide more guidance for the modeler in what to model than for instance UML but are still not focused on interoperability and thus employing them as-is would likely result in a lack of several aspects important for understanding the interoperability issues.

Recently several initiatives on interoperability have proposed interoperability frameworks to structure issues and concerns in the domain. Examples include The European Interoperability Framework in the eGovernment domain [10], the e-Health interoperability framework [11] and the Framework for Enterprise Interoperability [12]. These frameworks generally provide means to classify the interoperability

problems and solutions. At the same time they lack the ability to model interoperability situations and perform assessments of interoperability.

The ontology of interoperability (OoI) [13] is an approach towards a deeper understanding of interoperability. OoI prescribes a set of metamodels to describe interoperability from various viewpoints, including the communication metamodel aimed at describing interoperability situations. The language described in this article uses several of the concepts of OoI and additionally allows specific messaging situations to be modeled, something not offered by OoI. But foremost the OoI does not provide a means to assess the interoperability of the modeled scenario.

Several methods for assessing interoperability on a general scope have previously been suggested. The Levels of information Systems Interoperability (LISI) [14], Levels of Conceptual Interoperability Model (LCIM) [15] and i-Score [16]. Employing these methods would however require more domain knowledge in the field of interoperability than the assessment method presented in this paper. These methods have the same goal as the work presented in this paper, to assess interoperability. Different from the work presented here these methods are often based on maturity models as apposed to this approach where an probability of successful communication is derived.

3 Architecture Analysis

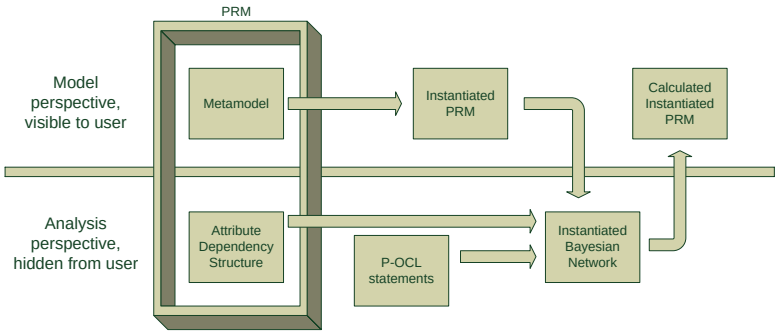

Fig. 1. Overview of the relationship between a PRM, P-OCL and Bayesian networks.

For the sake of architecture analysis it would be of great benefit to have a modeling language that also contains an evaluation mechanism. Furthermore, the ability of expressing uncertainty, both in the assessment theory as such and regarding the content of the model, would allow not only for an assessment to be performed, but also an indication of the precision in the analysis [17]. A *probabilistic relational model (PRM)* [4] specifies a template for a probability distribution over an architecture model. The template describes the metamodel *M* for the architecture model, and the probabilistic dependencies between attributes of the architecture objects. A PRM, together with an instantiated architecture model *I* of specific objects and relations, defines a probability distribution over the attributes of the objects. The

probability distribution can be used to infer the values of unknown attributes. This inference can also take into account evidence on the state of observed attributes.

A PRM Π specifies a probability distribution over all instantiations I of the metamodel M. As a Bayesian network it consists of a qualitative dependency structure, and associated quantitative parameters. The qualitative dependency structure is defined by associating attributes A of class X *(A.X)* with a set of parents *Pa(X.A),* where each parent is an attribute, either from the same class or another class in the metamodel related to X through the relationships of the metamodel. For example, the attribute *satisfied* of the class *Communication Need* may have as parent *CommunicationNeed.associatedTo.communicatesOver.isAvailable*, meaning that the probability that a certain communication need is satisfied depends on the probability that an appropriate message passing system is available. Note that a parent of an attribute may reference a set of attributes rather than a single one. In these cases, we let *X.A* depend probabilistically on an *aggregated* property over those attributes constructed using operations such and *AND, OR, MEAN* etc.

Considering the quantitative part of the PRM, given a set of parents for an attribute, we can define a local probability model by associating a conditional probability distribution with the attribute, *P(X.A |Pa(X.A))*. For instance, *P(CommunicationNeed.satisfied=True|MessagePassingSystem.isAvailable=False)= 10%* specifies the probability that communication need is satisfied, given the availability of the message passing system.

3.1.1 P-OCL

PRMs do not, however, provide any concise means to query the models for structural information such as “given two actors with a need to communicate, do these actors have a common language (modeled as a separate object)?” 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. [18]

This ability to query models would be of great benefit to interoperability analysis, since many interoperability problems are due to structural factors. OCL is, however, a deterministic language and thus incapable of leveraging the benefits of a probabilistic analysis as described above. This section briefly describes how OCL is extended to *Probabilistic* OCL or P-OCL for short. For a more comprehensive treatment see [5]. For the sake of the P-OCL analysis it is necessary to introduce an existence attribute E in all classes and relationships corresponding to the probability that the class or relationship exists.

From a black box perspective P-OCL is used in the same way as OCL, the P-OCL statements are very similar to those of OCL. The difference is that the result of a P-OCL statement is always assigned back to an attribute in the model, something generally not done with an OCL statement. The modeler must also specify evidence on the values for the existence attributes of the classes and relationships in the model. Given for instance a model of a person’s family and friends it is possible to evaluate whether a person’s father is friends with any of the fathers of the person’s friends with the following statement:

Self.parentalFriends := self.friend.father -> exist(self.father)

Using P-OCL, this evaluation will take into account the probability of the classes and relationships, i.e. does my friend really exist and, in this case more importantly, is he really my friend.

By combining the probability model of a PRM expressed in terms of parent attributes with P-OCL statements, cf. Fig. 1, it is possible to allow not only attributes to constitute the parents of an attribute but also various aspects pertaining to the structure of the model. This allows us to infer the probability that a certain attribute (e.g. *CommunicationNeed.satisfied*) assumes a specific value, given some evidence of the rest of the architecture instantiation. In this paper P-OCL statements are used for the main part of the analysis since many interoperability concerns are of the structural type. The general probability model of a PRM is instead used to aggregate attributes, e.g. the combination of the properties of a *CommunicationNeed* into the attribute *satisfied.*

4 A Probabilistic Relational Model for Interoperability Analysis

In this section a PRM for interoperability analysis is presented. The PRM is divided into two main parts, structural and conversation-specific, 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 other similar aspects.
- 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 to create architecture models amenable to analysis whereas the conversation-specific classes are a refinement requiring the structural aspects as well and allow for a more in-depth description and an interoperability analysis of a particular messaging situation.

This chapter is the main contribution of the paper and is outlined as follows: First, a brief overview of the PRM is provided for orientation purposes. Then, the classes, reference slots and attributes are described in more detail as well as the requirements that are needed in order to achieve interoperability.

4.1 Overview of the PRM

Several definitions of interoperability have been proposed and one of the most widely adopted is “the ability of two or more systems or components to exchange information and to use the information that has been exchanged” [2]. Adopting this view these systems or components (corresponding to the concept Actor used in this

paper) can be viewed as having a need for communication and the goal of the interoperability analysis is then to determine whether this need is satisfied. Actors can take various forms such as information systems, humans and whole enterprises but they all share the ability to actively operate on the information, e.g. interpreting or transforming information. To be able to exchange information, the actors need a medium for transmitting the information. Examples of such media, or Message-Passing Systems, are the Internet or Ethernet in computer communication, air in spoken communication between Actors of close distance or telephone lines when two parties use phones to communicate. Compared to Actors, the Message-Passing System is passive and can only transmit messages between Actors. Fig. 2 illustrates a simplified view of the PRM where the three concepts mentioned above correspond to the classes Communication Need, Actor and Message-Passing System respectively.

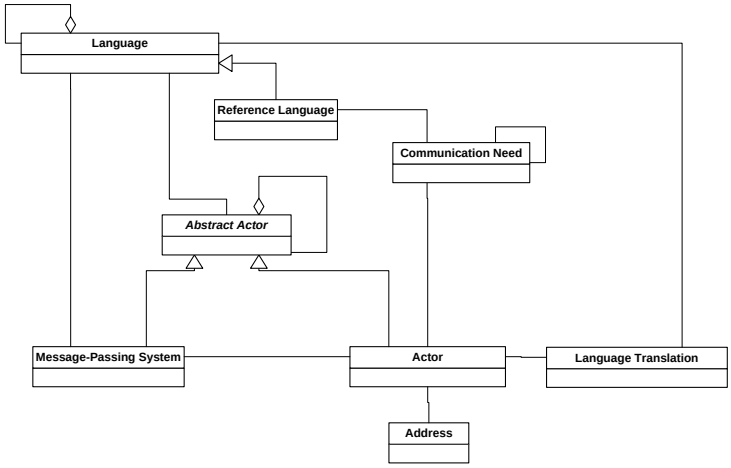

Fig. 2. Simplified PRM for the structural aspects of interoperability only showing the classes and slot chains.

Actors need to identify other actors to interoperate with. This is done using an Address. Furthermore the actors need to encode the information in a format, or Language, that the other party also is able to use. Examples of such Languages could be XML to be transmitted over media such as the Internet or spoken English in human communication. Actors can use several languages for encoding the communication but need to share at least one to communicate. Actors can translate between different Languages through Language Translations (since Languages can be sub-languages to each other it is sufficient to speak a Language higher up in the hierarchy). Message-passing systems also use Languages for transporting information (i.e. the protocol), such as HTTP for the Message-Passing System Internet.

A special Language is the Reference Language, a language in which the communication need can be evaluated. Reality is one possible Reference Language that is used if the Communication Need is concerned with altering the reality, e.g.

an enterprise receiving physical items from a supplier. Considering to 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. Most importantly, the reflexive aggregation relationship of the class Abstract Actor enables the modeler to describe the architecture on various levels of granularity, see the section Model Abstraction below.

In order to model specific conversations, four additional classes are needed. A Conversation Communication Need is the equivalent of the Communication Need and is shared by Actors. The goal of a Conversation Communication Need is to transmit a particular message between the actors, rather than expressing the probability that an unspecified message exchange will be successful as in the case for the previously described Communication Need. A Language can be detailed into its Constructs, defined as all valid symbols, words and sentences of the Language. A Conversation specifies a set of such constructs as the aim of the Actors to transmit. Constructs can be translated using Construct Translations and a set of such translations constitutes a Language Translation. Fig. 3 shows both the classes for structural and conversation-specific aspects and will be described in the remainder of this chapter.

4.2 Structural Aspects

The various model entities will now be described in greater detail. There are several requirements that need to be fulfilled so that a Communication Need can be *satisfied*. Firstly there must be a path between the Actors involved in a Communication Need. Furthermore, this path must be available; this is captured by the attributes *noPath* and *pathUnavailable* respectively. Requirements like these are expressed in P-OCL in order to be automatically assessed by an assessment tool. The definition of the *noPath* attribute is as follows in P-OCL:

context: ***Communication Need***
def: let
getNeighbors(a : Actor) : Set(Actor)
= ***a.MPS.Actor****->collect(****x : Actor*** *|* ***x****<>****a*** *and* ***getNeighbors(x))****-> asSet()->union(****a.MPS.Actor****)*
self.noPath *:= not* ***self.Actor****->forAll(****a1 : Actor*** *|* ***getNeighbors(self.Actor****->asSequence()->first())->exists(****a1****))*

where getNeighbours is a recursive support function needed in order to evaluate the *noPath* attribute in the latter part of the expression. In more detail, the Boolean *noPath* attribute is assigned the value False if all Actors of a Communication Need exist in the set returned by getNeighbors when called with one of the Actors of the Communication Need. The function getNeighbors receives one Actor as input and returns the union of all Actors related through the MPSs related to the input Actor and the recursive calls of the function which each of these Actors. Otherwise the attribute *noPath* is evaluated to True and this corresponds to evaluating if there is a communication path between the Actors. Furthermore, this evaluation of the structure of the model can be fully automated given an instantiated model. The remainder of

this paper will only express the requirements in natural language due to space restrictions, the full set of P-OCL expressions needed for the analysis can be found in [19].

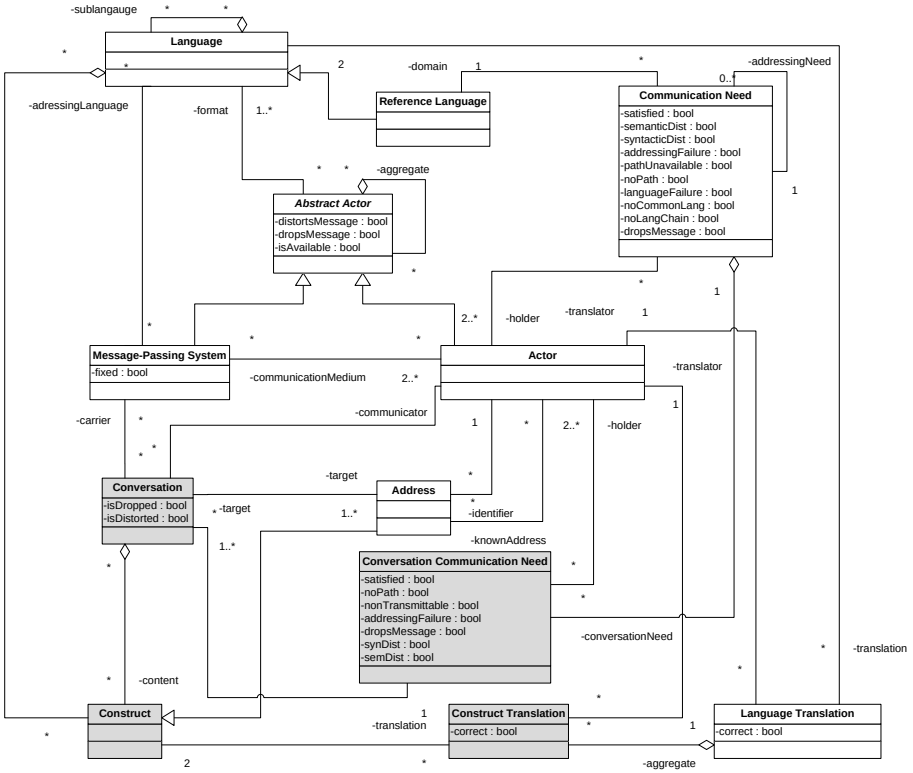

Fig. 3. The PRM for interoperability modeling and analysis containing the relevant classes and relationships for describing interoperability issues and performing interoperability assessments.

Returning to the requirements for satisfied communication needs, there must be a Language available that is spoken by all Actors which are related to this Communication Need, expressed through *noCommonLang*. Another requirement is that neither the syntax nor the semantics of the information which has been exchanged between the involved Actors has been modified so that it became unusable. This is modeled using the attributes *syntacticDistortion* and *semanticDistortion* respectively. It is also required that the addressing between the involved Actors is performed without errors, corresponding to the property *addressingFailure*. Finally, for a Communication Need to be satisfied, no messages should be dropped on the route from the sender to the receiver, the corresponding attribute used within the model is *dropsMessage*. Only if all of these potential sources of error are cleared out, a Communication Need can be satisfied, which is reflected in the attribute with the same name. Evaluating the conditions

described above is cumbersome but since they are defined in P-OCL [19] the evaluation can be automatically performed by a software tool.

The rest of the classes in the PRM will be described in four main groups. Firstly there are the classes that ensure a communication medium between the Actors sharing a Communication Need (the interoperation path). Secondly, classes related to language are described. This is followed by aspects covering addressing issues and finally the classes relating to a specific messaging situation, i.e. a conversation, are described.

4.2.1 Interoperation Path

Actors cannot be related directly to each other, there needs to be a transport medium that passes information between Actors. This medium is denoted Message Passing System and is related to Actors via the *communicationMedium* reference slot of the Actor. The class MPS is associated to the class Language through two reference slots. The first, *uses*, indicates the languages used for message passing, i.e. the protocol according to which messages are formatted while transported by the MPS. Secondly the *addessingLanguage* reference slot expresses how valid addresses are encoded on the MPS. For a Local Area Network MPS, the addressing Language could be IP addresses. MPSs can be separated into two categories. Firstly, MPSs in which it is not necessary to take care of the addressing, as it can be solved unambiguously because the number of involved parties is fixed. Secondly MPSs where the involved parties are not fixed, indicated by the attribute *fixed*. For example, the MPS is fixed if it represents a cable connecting two Actors directly; in such case the involved parties can be identified unambiguously, otherwise, e.g. using a network for communication, addressing is necessary.

Actors and MPSs are specializations of the class Abstract Actor, which gathers common properties and relationships of Actors and MPS. There exist three scenarios in which an Abstract Actor might impede a communication. Either the Abstract Actor loses information, it modifies data so that it becomes unusable or it is unable to take part in a communication because it is occupied or defect. These three possibilities are reflected in the three Attributes *dropsMessage*, *distortsMessage*, and *isAvailable* of the Abstract Actor class. A key feature of the Abstract Actor is the reflexive aggregation relationship that allows for abstraction in the models and thus modeling on various levels of detail, see the Model Abstraction section below.

4.2.2 Language

The languages that are used within a scenario are modeled as Language entities. To format according to a Language, however, only means that the Actor can perceive and distinguish the Constructs (see conversation-specific aspects below) of the language. To understand those Constructs, the Actor needs a Construct Translation, or its aggregate the Language Translation, ultimately to a Reference Language.

A Language might consist of several sub-languages. Being a sub-language means that the language can be fully expressed by the corresponding super-language, either

by being a subset of the super-language, e.g. HTML being a subset of XML with a set of specific tags, or by the super-language being a protocol for transmitting the sub-language, e.g. TCP being able to express, or rather transmit, XML (and thereby also HTML).

A Language can be mapped to other Languages, by the use of a Language Translation. 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 in.

4.2.3 Addressing

Actors need to be identified, which is done using Addresses such as an IP address on the Internet. The Actor has two reference slots to Address, *identifier* and *knownAddress*. The former is used for identification of an Actor whereas the latter constitutes the set of such identifiers known by a specific Actor. The Actors that communicate over a certain Message Passing System must provide the Address to the Message Passing System for correct delivery of the messages. It is therefore important that Actors describe their Addresses in a Language that is compatible with the addressing Language of the Message Passing System.

If direct knowledge of the needed addresses is missing, this barrier can be mitigated of an address broker, e.g. a DNS or UDDI. Such situations can be expressed as a separate Communication Need between the Actor that is lacking information and other Actors corresponding to the address broker. In the model this is expressed in terms of a submodel for these address communication needs that describe how addressing in the original communication need is achieved.

4.3 Conversation-specific Aspects

In addition to modeling the structural aspects described above it is also possible to model conversation-specific aspects. These aspects detail a particular message exchange between actors. Actors achieve ultimate communication success, i.e. interoperability, by transmitting the sought after conversation. For this purpose the classes Conversation, Conversation Communication Need, Construct and Construct Translation of Fig. 2 can be used for a more detailed scenario description.

A Language can be described in detail by its containing Constructs. Each possible message, which could be formulated in a language, is represented as a Construct. This is done to avoid the detailed modeling structure of the Language in terms of its grammar and thesaurus. During a message exchange Actors exchange several Constructs. This collection of transferred Constructs is called Conversation. As the modeling language is designed for static interoperability analysis, the order of the exchanged messages is not considered. A Conversation might be completely interrupted and not reach its destination. 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*. Conversations are sent through Message Passing Systems and are coupled with an Address, detailing the receiving Actor of a conversation.

For each specific conversation there is a Conversation Communication Need depicting the intension to exchange one instance of the Conversation class between a set of Actors. This class has a set of attributes similar to that of the Communication Need, as the requirements are similar for the two classes. There is one major difference, the attribute *nonTransmittable* ensuring that the Conversation is expressed in a language that can be passed between the Actors and over the Message Passing Systems employed in the scenario. Finally the class Construct Translation represents a mapping and transformation of one Construct of a Language to another one of another Language. A set of Construct Translations for constructs of a particular Language together form a Language Translation

4.4 Model Abstraction

When performing assessments based on models there is often a tradeoff between the cost of modeling and predictive power of the analysis based on the model. In order to facilitate this tradeoff, the aggregation relationship of the abstract actor enables the system to be described on various levels of granularity. As previously described, Actors can take various forms such as whole enterprises, departments or information systems. A coarser-grained actor, such as an enterprise, generally contains several more fine-grained actors, such as information systems and employees. The most abstract instantiation of the PRM would be a model of the structural aspects with one communication need, two actors connected to one message passing system and the languages of the included parties. Although such a model would be easy to comprehend, the actors and message passing systems would in most cases be large and complex, consisting of several other actors. This in turn makes it harder to assess the probability of message dropping, availability etc., resulting in a less credible interoperability analysis, i.e. less predictive power. On the other end of the scale, a very thorough modeling of all details would require a larger modeling effort and make the resulting model harder to understand but it would also enable a detailed interoperability analysis with high predictive power.

5 Example Usage

To illustrate the use of the model, the setting of an electric utility company is chosen. In Sweden, electric utilities have recently been by law mandated to bill customers based on their actual monthly consumption rather than on an estimate that is adjusted once a year. This has led to the introduction of automated meter reading systems and a need for communication between these systems and the billing system.

In this example we have a Communication Need ‘Get Meter Reading’ between the Actors ‘Billing System’ and ‘Meter Reading System’. These are connected using an ‘ISDN’ leased line of type Message Passing System that uses the protocol ‘X.25’ and the associated addressing language ‘X.121’. The ‘Meter Reading System’ uses two Languages, the common information model (‘CIM’)

and ‘EDIEL’ whereas the ‘Billing System’ only use ‘EDIEL’. Fig 4 depicts a model of this coarse-grained scenario including the sublanguage relationships between the languages. On this model it is then possible to perform interoperability analysis once the modeler provides a scenario-specific parameterization of the existence attributes of classes and relationships as well as the descriptive attributes of the Actor and Message Passing System.

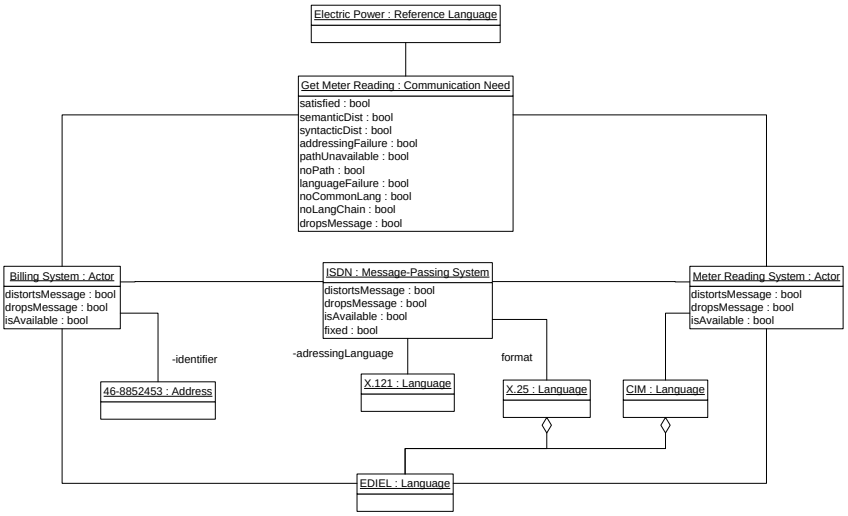

Fig. 4. Instantiated PRM for interoperability analysis of the communication need get meter reading. Reference slot names are only used when there is an ambiguity.

Returning to the requirements of the previous chapter, the P-OCL statements can be used to automate the analysis in a modeling tool and would for the current case show for instance: There is a path between the Actors of the Communication Need, these Actors do share a Language and this Language is transmittable on the MPS. It would however also show that the MPS requires addressing and at the same time the ‘Meter Reading System’ does not know the Address of the ‘Billing System’. Furthermore the model does not indicate the existence of an address broker, so addressing will constitute a problem (i.e. GetMeterReading.*addressingFailure* will be true). Having evaluated all the P-OCL statements, the dependency model of the PRM will be used in order to combine the attributes into the *satisfied* attribute, which due to the addressing problems will indicate that the Communication Need will not be *satisfied*. For more information on this aggregation, see [19].

6 Conclusions and Further Works

This article has demonstrated how PRMs and P-OCL statements can be employed for interoperability assessment. The contribution of the article is twofold. Firstly, the language for modeling interoperability scenarios, expressed in a PRM, provides a means to describe interoperability problems and solutions in a generic fashion.

Secondly, the P-OCL statements allow the modeler to assess the interoperability of the modeled scenario. A decision maker could employ this work for modeling various future scenarios and assess them with respect to interoperability.

The language was developed to be generic and capable of describing many different interoperability scenarios, not only in the information systems domain as outlined in the example above. A more specialized language would enable a more detailed analysis for that particular domain. Such specialization could be based on the language presented here and by PRM inheritance as described in [20]. At current, the language presented in this article is delimited to enabling factors for interoperability and does not cover directly preventive aspects such as various security measures. The versatility provided by the PRM formalism however allows such extensions to be added and it would be possible to create a new PRM covering both aspects.

Employing the language without tool support would be difficult for the decision makers it is intended for. In particular the evaluation of the P-OCL statements would be cumbersome. To aid in this, a software tool for enterprise architecture modeling and assessment based on PRMs [21] is currently being extended to handle P-OCL statements [22]. Using this tool, the decision maker can model current and future scenarios and automatically infer the degree of interoperability.

7 References

[1] Ullberg, J., Chen, D., Johnson, P. (2009). Barriers to Enterprise Interoperability. Proceedings of 2nd IFIP WG5.8 Workshop on Enterprise Interoperability (IWEI ´09). Springer Verlag

[2] IEEE (1990). Standard Glossary of Software Engineering Terminology. Std 610.12. New York: The Institute of Electrical and Electronics Engineers.

[3] Chen, D., Doumeingts, G., & Vernadat, F. (2008). Architectures for enterprise integration and interoperability: Past, present and future. Computers in Industry, 59(7), 647-659. doi: 10.1016/j.compind.2007.12.016.

[4] Getoor, L. Friedman, N. Koller, D. Pfeffer, A. and Taskar, B. Probabilistic relational models. MIT Press, 2007.

[5] Johnson, P. et al. (2011) p-OCL – a language for probabilistic inference of the structure in relational models. To be submitted

[6] Object Management Group (OMG). (2009). OMG Unified Modeling Language (OMG UML), Superstructure Version 2.2.

[7] Zachman, J.A. (1987). A Framework for Information Systems Architecture. IBM Systems Journal, 26(3),454-470.

[8] Department of Defense Architecture Framework Working Group. (2004). DoD Architecture Framework, version 1.0. Department of Defense, USA, 2004

[9] Lankhorst M. et al. (2005). Enterprise Architecture At Work. Berlin, Heidelberg: Springer Verlag.

[10] IDABC, Enterprise and Industry DG. (2004). *European interoperability framework for pan-European egovernment services.* version 1.0, Brussels.

[11] National E-Health Transition Authority (NEHTA). (2007). Interoperability Framework, Version 2.0. Sydney: National E-Health Transition Authority.

[12] Chen, D. & Daclin, N. (2006). Framework for Enterprise Interoperability. In EI2N, 2nd International Workshop on Enterprise Integration, Interoperability and Networking.

[13] Ruokolainen, T., Naudet, Y. & Latour. T. (2007). An ontology of interoperability in inter-enterprise communities. In proceedings of Interoperability for Enterprise Software and Applications, I-ESA'07.

[14] Kasunic, M. & Anderson, W. (2004). Measuring Systems Interoperability: Challenges and Opportunities. Technical Note, CMU/SEI-2004-TN-003, Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

[15] Tolk, A. & Muguira, J. (2003). The Levels of Conceptual Interoperability Model. Proceeding s of the 2003 Fall Simulation Interoperability Workshop.

[16] Ford, T., Colombi, J., Graham, S. & Jacques, D. (2007). The Interoperability Score. Proceedings of the Fifth Annual Conference on Systems Engineering Research.

[17] Johnson, P., Lagerström, R., Närman, P., Simonsson, M. (2007) Enterprise Architecture Analysis with Extended Influence Diagrams, Information Systems Frontiers. 9(2).

[18] Object Management Group (OMG). (2006) Object Constraint Language specification, version 2.0 formal/06-05-01

[19] Ullberg, J. (2010) P-OCL expressions for interoperability analysis, http://www.ics.kth.se/POCL/interoperability.html

[20] Sommestad, T., Ekstedt, M. & Johnson, P. (2010) A Probabilistic Relational Model for Security Risk Analysis, Computers & Security.

[21] Buschle, M. Ullberg, J., Franke, U., Lagerström, R., Sommestad, T. (2010) A Tool for Enterprise Architecture Analysis using the PRM formalism. Forum of The 22nd International Conference on Advanced Information Systems Engineering, CAiSE2010 Forum PostProceedings.

[22] Ullberg, J., Franke, U., Buschle, M. & Johnson, P. (2010). A Tool for Interoperability Analysis of Enterprise Architecture Models using Pi-OCL. Proceedings of The international conference on Interoperability for Enterprise Software and Applications (I-ESA'10).