A Tool for Enterprise Architecture Analysis using the PRM formalism

Markus Buschle, Johan Ullberg, Ulrik Franke, Robert Lagerström, and Teodor Sommestad

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

Abstract. Enterprise architecture advocates for model-based decision-making on enterprise-wide information system issues. In order to provide decision-making support, enterprise architecture models should not only be descriptive but also enable analysis. This paper presents a software tool, currently under development, for the evaluation of enterprise architecture models. In particular, the paper focuses on how to encode scientific theories so that they can be used for model-based analysis and reasoning under uncertainty. The tool architecture is described, and a case study shows how the tool supports the process of enterprise architecture analysis.

Keywords: Enterprise Architecture, Probabilistic relational Models, Software tool, Security Analysis

1 Introduction

Over the last two decades, enterprise architecture has grown into an established approach for holistic management of information systems in organizations [1, 2]. A number of enterprise architecture initiatives have been proposed, such as The Open Group Architecture Framework (TOGAF) [3], the Zachman Framework [4], and military architectural frameworks such as DoDAF [5] and NAF [6]. The core concept of the enterprise architecture approach is the employment of models, in terms of diagrammatic descriptions of information systems and their environment. Diagrammatic descriptions of IT systems and their environment are heavily used. However, enterprise architecture models are not limited to descriptive use only, but can also be employed to predict the behavior and effects of decisions. Rather than modifying enterprise information systems using trial and error, models allow predictions about the behavior of future architectures.

One prominent challenge to rational decision making is uncertainty. Therefore, a good enterprise architecture model should be able to capture uncertainties about assessment theory, system configuration or data quality, thus providing better decision support and risk management.

What constitutes a ”good” enterprise architecture model is dependent on its purpose, i.e. the type of analysis it is intended to support [7]. For instance in the case of analyzing cyber security, the property of whether it is possible to reconfigure a firewall is of interest. This property however, is irrelevant for a number of other analyses, such as performance evaluation or data quality analysis.

Several enterprise architecture software tools are available on the market, including Metis [8], System Architect [9] and Aris [10]. These tools generally focus on the modeling of an architecture whereas the analysis functionality is generally limited to performing an inventory or to sum costs over the modeled architecture. None of the mentioned tools has significant capabilities for system quality analysis based on an elaborated theory. Furthermore, these tools do not support the consideration of uncertainty as described above.

In this paper an enterprise architecture software tool is presented. This tool does not only provide functionality to model enterprise architectures, but also supports the analysis of them. In order to support enterprise architecture analysis as it has been outlined in [7] the tool consists of two main components. In the first component the theory relevant to analyze a certain system quality, such as data quality or modifiability, is modeled. One can consider this as the definition of a language tailored to describe a certain aspect, e.g. cyber security. The second component supports the application of the theory to evaluate a specific enterprise architecture. This is done by modeling the ”as-is” or ”to-be” architecture of the enterprise. Based on the created models it is possible to determine how the architecture fulfills the requirements as they have been defined in the theory. The two-component architecture encourages the reuse of the developed theory as it is possible to use the same language to describe several architecture instances. The presented tool makes use of the Probabilistic Relation Models (PRM) formalism as it has been presented in [11] and can thereby manage the uncertainty aspects discussed above.

2 Enterprise Architecture Analysis

Enterprise architecture models serve several purposes. Kurpjuweit and Winter [12] identify three distinct modeling purposes with regard to information systems, viz. (i) documentation and communication, (ii) analysis and explanation and (iii) design. The present article focuses on the analysis and explanation (which is not to denigrate the usefulness of the others). The reason is that analysis and explanation are closely related to the notion of proper *goals* for enterprise architecture efforts. For example, a business goal of decreasing downtime costs immediately leads to an analysis interest in availability. This, in turn, defines the modeling needs, e.g. the need to collect data on mean times to failure and repair. In this sense, analysis is at the core of making rational decisions about information systems [7] [13]. An analysis-centric process of enterprise architecture is illustrated in Fig. 1. In the first step, assessment scoping, the problem is described in terms of one or a set of potential future scenarios of the enterprise and in terms of the assessment criteria with its theory (the PRM in the figure) to be used for scenario evaluation. In the second step, the scenarios are detailed by a process of evidence collection, resulting in a model (instantiated PRM, in the figure) for each scenario. In the final step, analysis, quantitative values of the models' quality attributes are calculated and the results are then visualized in the form of e.g. enterprise architecture diagrams.

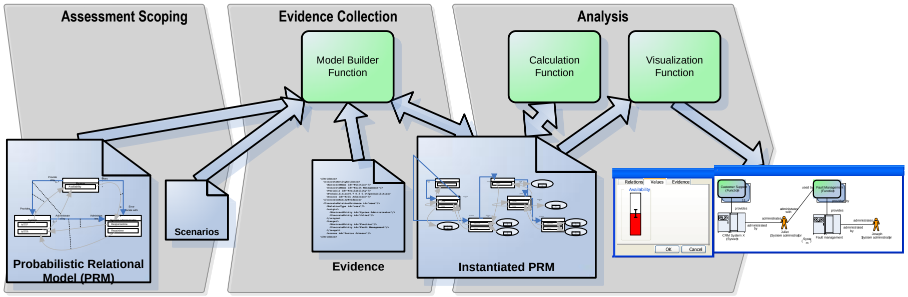

Fig. 1. The process of enterprise architecture analysis with three main activities: (i) setting the goal, (ii) collecting evidence and (iii) performing the analysis.

More concretely, assume that a decision maker in an electric utility is contemplating changes related to the configuration of a substation. The modification of a new access control policy would reduce the probability that someone installs malware on a system and thereby reduce the risk that this type of unwanted software is executed. The question for the decision maker is whether this change is feasible or not.

As mentioned in the first step *assessment scoping* the decision maker identifies the available decision alternatives, i.e. the enterprise information system scenarios. In this step, the decision maker also needs to determine how the scenario should be evaluated, i.e. the goal of the assessment. One such goal could be to assess the security [14] of an information system. Other goals could be to assess the availability [15], interoperability [16] or data quality [17] [18] of the proposed to-be architecture. Often several quality attributes are desirable goals. In this paper, without loss of generality, we simplify the problem to the assessment of security of an electric powerstation.

Information about the involved systems and their organizational context is required for a good understanding of their data quality. For instance, it is reasonable to believe that a firewall would increase the probability that the system is secure. The availability of the firewall is thus one factor that can affect the security and should therefore be recorded in the scenario model. The decision maker needs to understand what information to gather and also ensure that this information is indeed collected and modeled. Overall, the effort aims to understand which attributes causally influence the selected goal, viz. data quality. It might happen that the attributes identified do not directly influence the goal. If so, an iterative approach can be employed to identify further attributes causally affecting the attributes found in the previous iteration. This iterative process continues until all paths of attributes and causal relations between them, have been broken down into attributes that are directly controllable for the decision maker [19] (cf. Fig. 2).

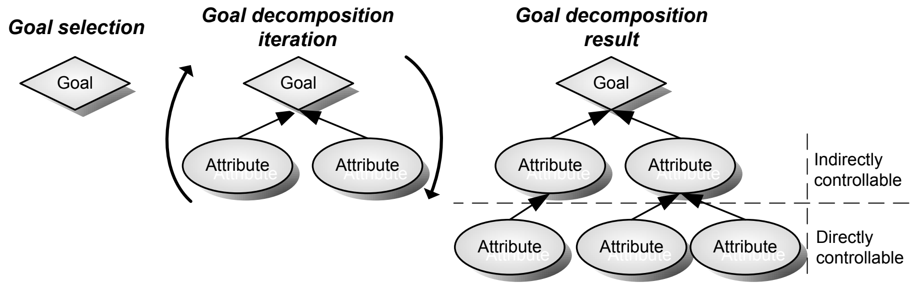

Fig. 2. . Goal decomposition method, from [19].

In the second step *collecting evidence* the scenarios need to be detailed with actual information to facilitate their analysis of them. Thus, once the appropriate attributes have been set, the corresponding data is collected throughout the organization. In particular, it should be noted here that the collected data will not be perfect. Rather, it risks being incomplete and uncertain. The tool handles this by allowing the user to enter the credibility of the evidence depending on how large the deviations from the true value are judged to be.

In the third and final step, *performing the analysis*, the decision alternatives are analyzed with respect to the goal set e.g. security. The mathematical formalism to be presented in section 2.1 plays a vital role in this analysis. Using conditional probabilities and Bayes' rule, it is possible to infer the values of the variables in the goal decomposition under different architecture scenarios [20]. By using the PRM formalism, the architecture analysis accounts for two kinds of potential uncertainties: that of the attribute values as well as that of the causal relations as such. Using this analysis framework, the pros and cons of the scenarios can be weighted against each other in order to determine which alternative ought to be preferred.

2.1 Probabilistic Relational Models

A *probabilistic relational model* (PRM) [21] specifies a template for a probability distribution over an architecture model. The template describes the metamodel for the architecture model, and the probabilistic dependencies between attributes of the architecture objects. A PRM, together with an instantiated architecture model 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, given evidence of the values of a set of known attributes. PRMs are related to Bayesian Networks. As it is succinctly put in [11], PRMs "are to Bayesian networks as relational logic is to propositional logic".

A PRM model may be instantiated as a *relational skeleton*, σ_r, containing objects, object relationships, and attributes. Furthermore, a *qualitative dependency structure* $\mathcal{S}$ defines the details of the attribute relationships, i.e. the sets of probabilistic parents influencing each attribute. Finally, the PRM is completed by the set of *parameters* $\theta_{\mathcal{S}}$ specifying the full conditional probabilistic dependencies between attributes in the form of numbers in Conditional Probability Matrices (CPM). The following expression thus defines the conditional probability of an instance $\mathcal{I}$, given σ_r, $\mathcal{S}$, and $\theta_{\mathcal{S}}$:

$$P(\mathcal{I}|\sigma_r, \mathcal{S}, \theta_{\mathcal{S}}) = \prod_{x \in \sigma_r} \prod_{A \in \mathcal{A}(x)} P(\mathcal{I}_{x.A}|\mathcal{I}_{Pa(x.A)})$$

$$= \prod_{X_i} \prod_{A \in \mathcal{A}(X_i)} \prod_{x \in \sigma_r(X_i)} P(\mathcal{I}_{x.A}|\mathcal{I}_{Pa(x.A)})$$

Compared to the standard chain rule for Bayesian networks, this equation is different in three ways: (i) the random variables are the attributes of a set of objects, (ii) the parents of a random variable depend on the model context of the object, and (iii) the parameters are shared between the attributes of objects in the same class. In other words, the variables in the dependency structure are the properties of the objects in the instantiated information model, and their causal relations are expressed by the CPM [11].

A PRM thus constitutes a formal machinery for calculating the probabilities of various architecture instantiations. This allows us to infer the probability that a certain attribute assumes a specific value, given some (possibly incomplete) evidence of the rest of the architecture instantiation. In addition to expressing and inferring uncertainty about attribute values as specified above, PRMs also provide support for specifying uncertainty about the structure of the instantiations.

PRMs additionally allow specializing classes through inheritance relationships. Classes can be related to each other using the subclass relation $\prec$, and each class X is associated with a finite set of subclasses C[X]. So if $Z, Y \in C[X]$, both Z and Y are subclasses of X. If $Z \prec Y$ then Z is a subclass of Y, and vice versa Y is a superclass of Z. A subclass Z always contains all dependencies and attributes of its superclass Y. PRMs also allow the dependencies and conditional probability distributions of inherited attributes to be specialized in subclasses.

3 Architecture of the Tool

The presented tool is implemented in Java and is based on a Model-View-Controller architecture [22] [23]. Where the model represents the knowledge and considered data that in this case are PRM and instantiated PRM respectively. The implementation contains a mapping of the PRM structure to JAVA-classes. Thereby each part of a PRM (e.g. PRM-classes, their attributes, and their relations) can be used and combined as regular JAVA-objects. The view is in charge of the visualization of the model for the tool user. It presents the considered data in an understandable way. The view is updated as soon as the model changes.

Finally the controller links the user to the application making the program react on the users input. Changes of the model are perforemd via the controller that acts as an interface in this case. Thereby a decoupling of data access, program logic, data presentation, and user interaction can be ensured. This facilitates the extension of the implementation and eliminates potential error sources as the program code is structured and functionality grouped according to its purpose.

The data model for PRMs and instantiated PRMs, is specified in XSD and stored in XML files [24]. This is done through an application of the Castor library [25]. As XML is a widespread format created models can be imported into other applications and data does not need to be captured a second time. The user interface is built upon the NetBeans Visual Library [26] with usage of the JApplication framework. This library provides a set of reusable components, called widgets, and can be applied to create visualization. The widgets can be aggregated and related to each other thereby reflecting the creation of models intuitively. These models are drawn on a special canvas that in the NetBeans terminology is called scene. Besides the modeling capabilities the user interface provides the one applying the tool with support functionalities such as filtering, tagging, and exporting. These well-defined tasks are performed by corresponding tailored services that are implemented following the singleton pattern to ensure data consistency. The architecture described is depicted in Fig. 3, whereas how the user interface (and thereby the actions performed by the user) is linked to the architecture us show in Fig. 4.

The tool is separated into two units, one supporting the modeling of the PRM, the other one makes the tool user able to instantiate and analyze this defined structure. These parts have to be used sequentially, reflecting the method that has been described in section 2, starting with the modeling of classes and their attributes as well as the relationships and dependencies between them. Thereby the focus of the analysis is set, as the defined classes reflect the domain of interest. The second component of the Enterprise Architecture Analysis Tool (EAT) allows the instantiation of the PRM. Thereby one or several scenarios of interests are modeled according to constraints defined in the dependency structure for the PRM. Afterwards the analysis is performed. Therefore the instantiated PRM is translated into a Bayesian network that is understandable by the Smile library [27]. This library performs the evaluation of the network. Finally the calculated values are written back to the instantiated PRM and visualized for the tool user.

The person applying the tool can then compare the modeled scenarios and their contained classes by considering the probabilities that attributes of the classes are in a certain state; thereby the identification of the configuration that qualifies best is made possible.

4 Example Tool Application

This section will illustrate how the tool can be applied in practice. The metamodel and instance model presented here are drawn from a case study performed

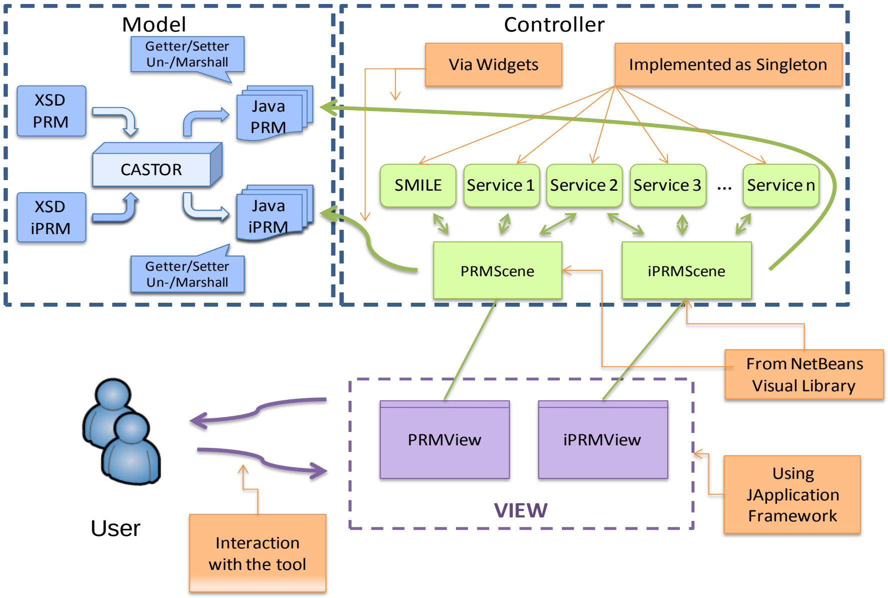

Fig. 3. High level architecture of the tool illustrating the main components. "iPRM" is shorthand notation for instantiated PRM.

at a Swedish power utility company in November 2009. In this case study the cyber security of an electric substation was the concern. The metamodel is thus intended to support cyber security analysis and the instance model represents one of the utility's substations. The qualitative structure of the metamodel (or PRM) was created based on a literature review; the quantitative part was in this case assessed (subjectively) by a security researcher. Interviews with a system administrator and investigations of system documentation were used to create the instance model.

4.1 Probabilistic relational model over security

The PRM used in this case study is depicted in Fig. 5. The PRM covers a number of concepts that are of relevance to the cyber security computer networks including firewalls, data flows, software services network zones and organizational functions. The qualitative structure of this PRM is described below together with some examples of conditional probabilities defined in the PRM.

The primary purpose of firewalls is to control access to network addresses. They do so by blocking unwanted data flows from adjacent zones, and by allowing those that are wanted. With a protection scheme following the principle 'deny by default', a *Firewall* will allow a number of *DataFlows* to pass into the secure *Zone* from other *Zones*. In this ConcretePRM a *Firewall* holds the reference

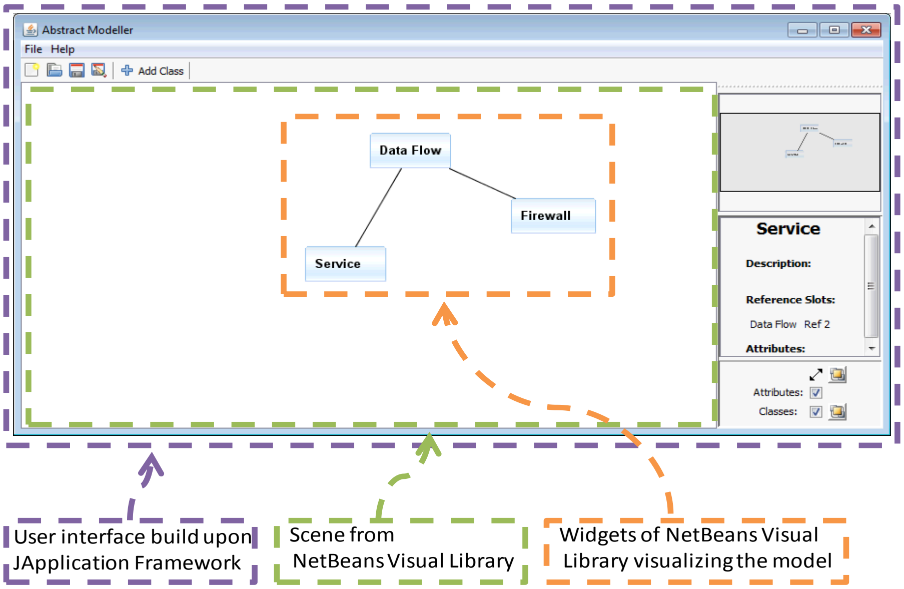

Fig. 4. User interface of the tool illustrating how the user interacts with the tool. Colors of boxes match with concepts described in Fig. 3.

slots *Allow* with range *DataFlow* which point to data flows that are allowed. A Firewall also has the reference slots *SecureZone* and *InsecureZone* with range *Zone* which refers to the zones that are directly separated by the firewall.

The *Firewall* has the attribute *PossibleToReconfigure* which indicates if it possible for a threat agent to reconfigure the firewall or not. This attribute influences the attribute *Firewall.Functioning* which indicates whether the firewall functions as it should. If the firewall is working it will prevent unauthorized connections from insecure zones to secure zones. The attribute *Service.PossibleToConnect* states whether the threat agent can connect to a *Service*. The threat agent can also connect to the service if it has access to the service's zone or if data flows are allowed from a zone where the threat agent have access. If the threat agent has access or not is expressed through the attribute *Zone.PossibleToAccess* which has a different value in the subclasses *PublicNetwork*, *CorportateNetwork* and *ProcessNetwork*.

If it is possible to connect to the service (i.e. Service.PossibleToConnect=True) it might be possible to exploit a vulnerability in the service. The attribute *Service.PossibleToExploitVulnerability* expresses whether this is possible or not. The reference slot *Service.Host* points to the *Host* that executes the service. The possibility to exploit vulnerabilities in the service influences if it possible to execute malware on the service's host. The possibility to execute malware on the host is also influenced by the existence of a functioning malware scanner in the host. The

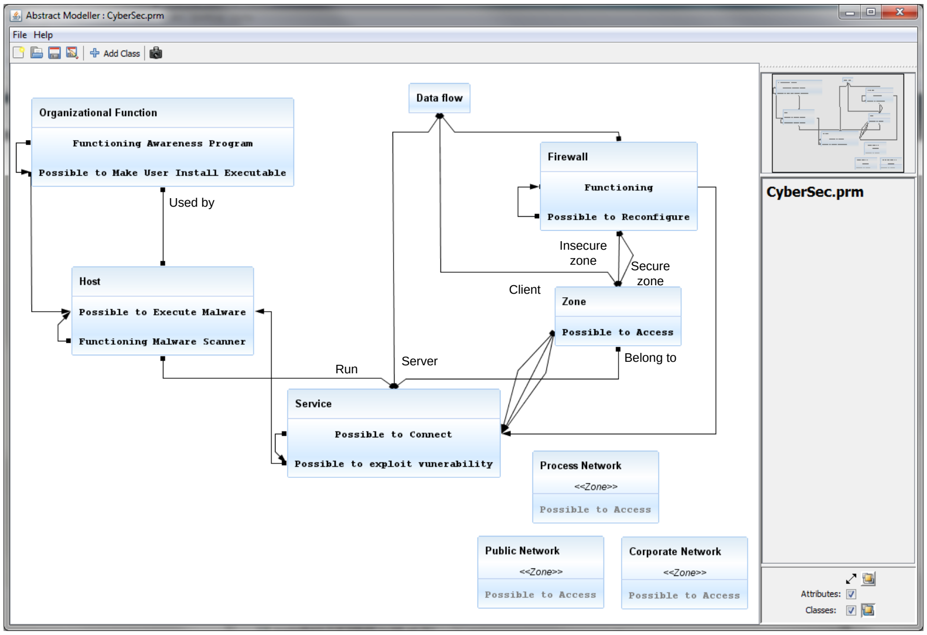

Fig. 5. The PRM for cyber security analysis showing classes and attributes relevant for the analysis.

attribute *Host.FunctioningMalwareScanner* indicates whether this is the case or not. Another way to influence the possibility to execute malware is through users in the *Organizational Function* that use the host and that install executables, i.e. the attribute *OrganizationalFunction.MakeUserInstallExecutable*.

4.2 Instance Model

The classes and reference slots in the PRM were used to model one of the utility's substations (cf. Fig. 6 for a screenshot of the tool). Four network zones where found in this study.

The *OfficeNetwork* is the insecure side of the *CorporateFirewall*. The *CorporateFirewall* allows two data flows: *RemoteDesktop* to pass through from the *OfficeNetwork* to the service *TerminalServices* and the data flow *SubstationCommunication* from the zone *ControlCenterNetwork* to *ControlSystemServer*. The *GatewayFirewall* is connected to the Internet and allows data to pass from both the *OfficeNetwork* and the *SuppliersLAN*.

Within the substation there are two instances of the *ProcessNetwork*. The *SubstationLAN* is where the services *ControlSystemServer* and *TerminalServices* belong; the service *VNCInterface* belongs to the *ModemLAN*. The *ControlSystemServer* is within the host *StationController*. The *ControlSystemServer* is also the server side of the data flow *SubstationCommunication*.

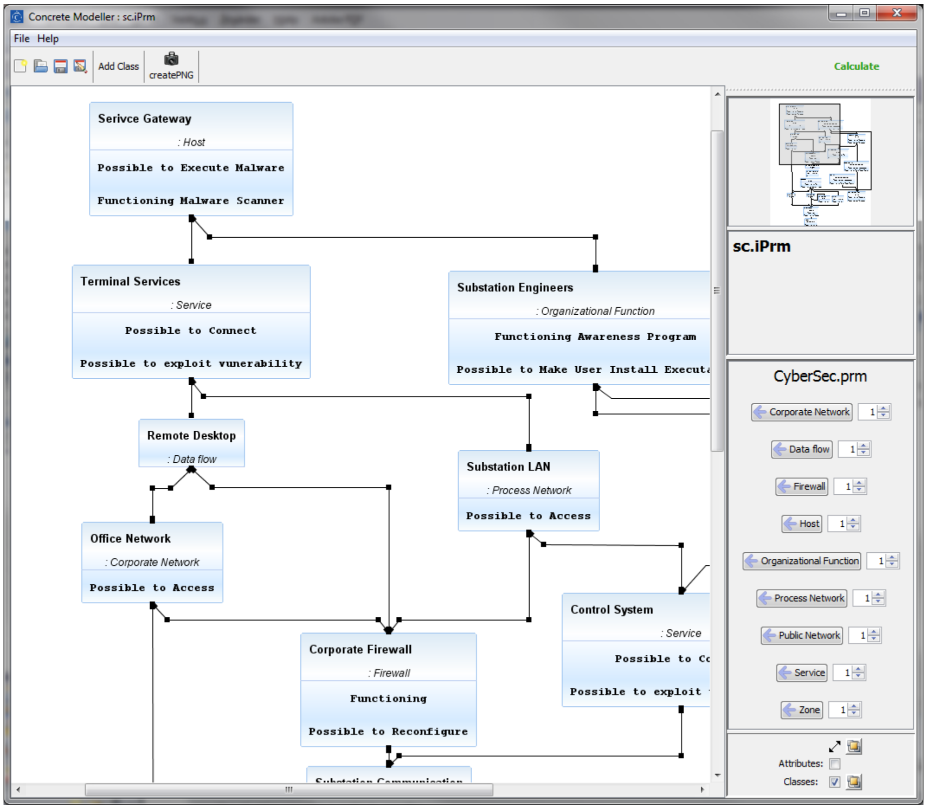

Fig. 6. Screen shot illustrating part of the instantiated PRM for cyber security analysis. For readability only the classes and reference slots are shown in the picture, attribute relationships are hidden.

The service *TerminalServices* is associated with the host *ServiceGateway* and acts as the server side of the data flow *RemoteDesktop*. The *ServiceGateway* is known to have a malware scanner that is functioning and evidence to support this fact is stored within the model. The service *VNCInterface* belongs to the *ModemLAN* and is executed by the host *EmbeddedController*.

Three organizational functions use hosts within the substation: *Supplier*, *FieldEngineers* and *SubstationEngineers*. The *EmbeddedController* is used by two organizational functions: *Supplier* and *FieldEngineers*. The *ServiceGateway* is used solely by the *SubstationEngineers* and the *StationController* is used by both *SubstationEngineers* and *FieldEngineers*. None of the organizational functions are covered by a functioning awareness program, i.e. *FunctioningAwarenessProgram=false* for all instances of *OrganizationalFunction*. The complete PRM is shown in Fig. 7

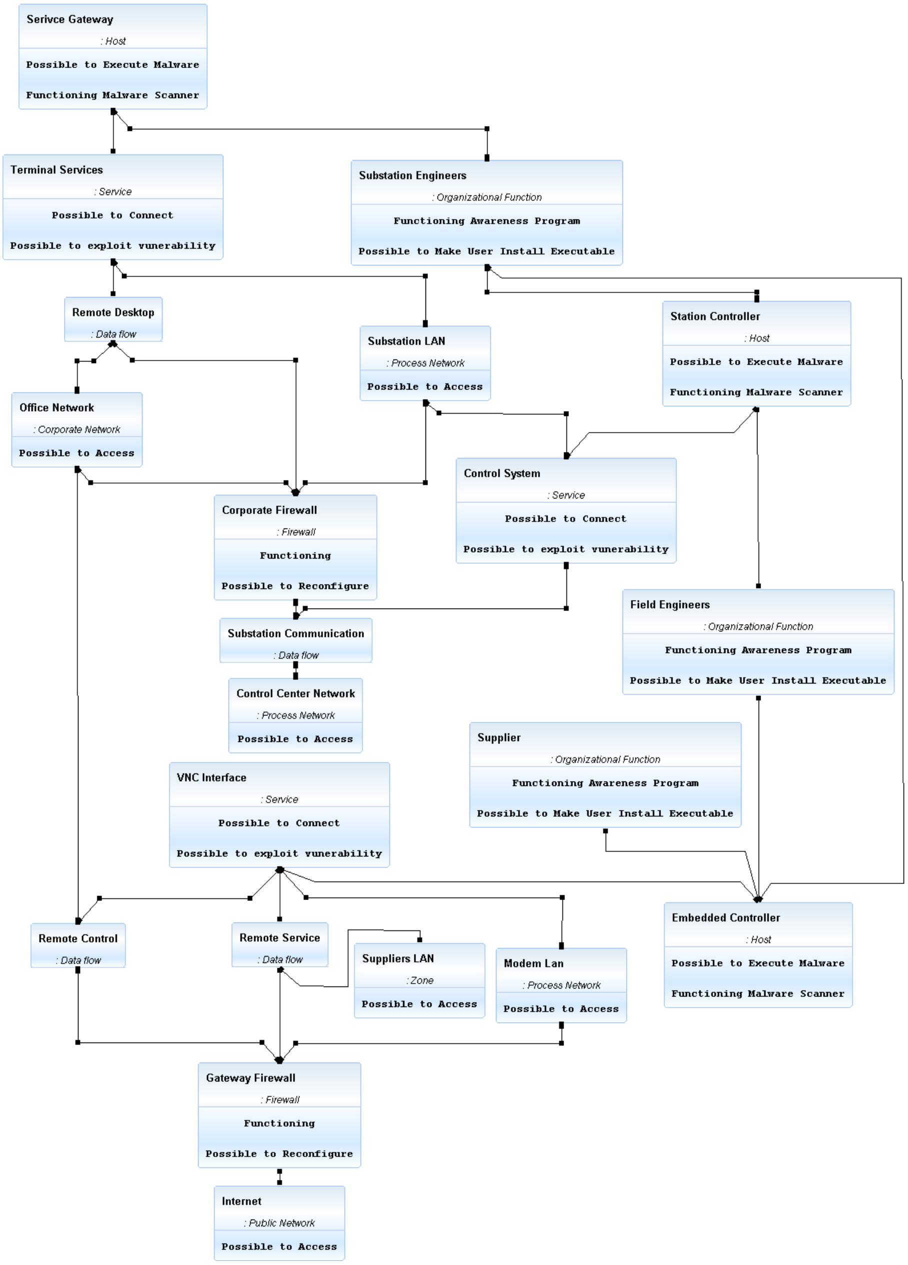

Fig. 7. Instantiated PRM for cyber security analysis. For readability only the classes and reference slots are shown in the picture, attribute relationships are hidden.

4.3 Scenario analysis

The instance model shown in Fig. 6 represents the architecture that existed at the time of the assessment. With this architecture as a starting point, different alternative scenarios were assessed.

One such scenario was to introduce an awareness program for substation engineers, i.e. to change *SubstationEngineers.FunctioningAwarnessProgram* to *True*. The impact of this is calculated to change *SubstationEngineers.MakeUserInstall Executable* from 20 % to 10 %, which in turn influences the *PossibleToExecuteMalware*-attribute in the hosts *StationController*, *EmbeddedController* and *ServiceGateway*.

Another scenario that was investigated was the impact of a new access control policy. This modification would ensure that *SubstationEngineers* do not use the *EmbeddedController*. A change like this would remove the dependency relationship between *SubstationEngineers.MakeUserInstallExecutable* and *EmbeddedController.PossibleToExecuteMalware*. For all cases other than *SubstationEngineers.MakeUserInstallExecutable=False* this would reduce the probability of *EmbeddedController.PossibleToExecuteMalware* being *true*. Making the change in the tool with the present PRM changes *P(EmbeddedController. PossibleToExecuteMalware)* from 50 % to 37 %.

5 Discussion and Future works

This paper presents a tool which supports enterprise architecture analysis with the use of the PRM formalism. While providing a powerful mechanism for the use of discrete variables in an analysis, the PRM formalism in its initial form has a few weaknesses that deserve further studies. Several system qualities are typically analyzed through the usage of continuous variables e.g. in [28] continuous variables are used for performance analysis. In order to perform those evaluations with support of the presented tool it is necessary to discretize all continuous variables. At the moment we are investigating how the PRM formalism can be extended so that it can be used with combinations of continuous and discrete variables, so called hybrid networks [29], as well as a corresponding tool implementation.

Another weakness of the PRM formalism is that it does not provide any means to query the models for structural information such as "given an information system, how many elements does the set of related data objects contain?" The Object Constraint Language (OCL) [30] is a formal language developed to describe constraints on UML models. OCL provides a means to specify such constraints and perform queries on the models in a formal language. OCL in its original form is side effect free, but currently an imperative version of OCL is being added to the tool. Thereby the analysis functionality can be extended to consider the structure of the PRM instantiation more comprehensively.

Besides the two mentioned shortcomings of the formalism used there are some improvements with respect to usability. Regarding the user interface of the tool, we are planning to make the models more intuitive and the information provided easier to understand. Enterprise architecture models are more understandable if they only depict the interesting parts of the model (in a goal-sense). Therefore, the tool should be extended to support views and viewpoints, e.g. as presented in [28]. Additionally we plan the support of iconic visualization of typical enterprise architecture elements, such as applications or data objects, to present the models in an easily understandable way. Finally we are planning to integrate the support of predefined model components. As models based on the same metamodel are likely to have common parts, the modeling process can be sped up if common building blocks are offered by the metamodel provider and used by the person that creates a certain model.

6 Conclusion

In this paper a tool and method for analysis of enterprise architecture scenarios was presented. To fulfill this purpose the tool consists of two separate parts, one for defining analysis theory and another for enterprise architecture modeling, and makes use of the PRM formalism for specifying theory. Applying this formalism allows for the consideration of uncertainty, an aspect that so far is uncommon in the field of enterprise architecture analysis. The paper describes the PRM formalism as well as the underlying architecture of the tool briefly.

In the paper an example of security assessment was outlined, but the tool supports the analysis of various quality attributes such as maintainability, data quality, and interoperability. The tool supports information system decision making as it allows the comparison of several scenarios with regard to a system quality. Thereby the ”as-is” as well as several ”to-be” architecture of an enterprise can be compared quantitatively in order to find the one that best satisfies decision maker requirements.

References

1. Ross, J.W., Weill, P., Robertson, D.: Enterprise Architecture As Strategy: Creating a Foundation for Business Execution. Harvard Business School Press (August 2006)
2. Winter, R., Fischer, R.: Essential layers, artifacts, and dependencies of enterprise architecture. Journal of Enterprise Architecture **Volume 3, Number 2** (2007) 7–18
3. The Open Group: TOGAF 2007 edition. Van Haren Publishing, Zaltbommel, Netherlands (2008)
4. Zachman, J.A.: A framework for information systems architecture. IBM Syst. J. **26**(3) (1987) 276–292
5. Department of Defense Architecture Framework Working Group: DoD Architecture Framework, version 1.5. Technical report, Department of Defense, USA (2007)
6. NAF: NATO C3 Technical Architecture. (2005)
7. Johnson, P., Ekstedt, M.: Enterprise Architecture – Models and Analyses for Information Systems Decision Making. Studentlitteratur, Sweden (2007)
8. Troux Technologies: Metis. `http://www.troux.com/products/` (March 2010)

9. IBM: System Architect. http://www-01.ibm.com/software/awdtools/systemarchitect/productline/ (March 2010)
10. Scheer, A.: Business process engineering: Reference models for industrial enterprises. Springer-Verlag New York, Inc. Secaucus, NJ, USA (1994)
11. Getoor, L., Friedman, N., Koller, D., Pfeffer, A., Taskar, B.: Probabilistic relational models. In Getoor, L., Taskar, B., eds.: An Introduction to Statistical Relational Learning. MIT Press (2007)
12. Kurpjuweit, S., Winter, R.: Viewpoint-based meta model engineering. In: Enterprise Modelling and Information Systems Architectures (EMISA 2007) (2007)
13. Iacob, M., Jonkers, H.: Quantitative analysis of enterprise architectures. Interoperability of Enterprise Software and Applications (2006) 239–252
14. Sommestad, T., Ekstedt, M., Johnson, P.: A probabilistic relational model for security risk analysis. Computers & Security (February 2010) Accepted.
15. Franke, U., Johnson, P., König, J., Marcks von Würtemberg, L.: Availability of enterprise IT systems – an expert-based bayesian model. In: Proc. Fourth International Workshop on Software Quality and Maintainability (WSQM 2010), Madrid. (March 2010)
16. Ullberg, J., Lagerström, R., Johnson, P.: A framework for service interoperability analysis using enterprise architecture models. In: IEEE International Conference on Services Computing. (July 2008)
17. Redman, T.: Data quality for the information age. Artech House, Inc. Norwood, MA, USA (1997)
18. Redman, T.: Data quality: the field guide. Digital Pr (2001)
19. Lagerström, R., Saat, J., Franke, U., Aier, S., Ekstedt, M.: Enterprise meta modeling methods – combining a stakeholder-oriented and a causality-based approach. In: Enterprise, Business-Process and Information Systems Modeling, Lecture Notes in Business Information Processing. Volume 29., Springer Berlin Heidelberg (June 2009) 381–393 ISSN 1865-1348.
20. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2001)
21. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: Proc. of the 16th International Joint Conference on Artificial Intelligence, Morgan Kaufman (1999) 1300–1309
22. Reenskaug, T.: Models-views-controllers. Technical note, Xerox PARC (1979)
23. Sun Microsystems: Design Pattern: Model-View-Controller. http://java.sun.com/blueprints/patterns/MVC.html (2002)
24. Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., Yergeau, F.: Extensible markup language (XML) 1.0. W3C recommendation **6** (2000)
25. ExoLab Group: The Castor Project. http://www.castor.org/ (March 2010)
26. NetBeans: NetBeans Visual Library. http://graph.netbeans.org (March 2010)
27. Decision Systems Laboratory of the University of Pittsburgh: SMILE. http://genie.sis.pitt.edu/ (March 2010)
28. Lankhorst, M.: Enterprise architecture at work : modelling, communication, and analysis. Springer (2005)
29. Lauritzen, S.: Propagation of probabilities, means, and variances in mixed graphical association models. Journal of the American Statistical Association **87**(420) (1992) 1098–1108
30. Object Management Group: Object Constraint Language specification, version 2.0 formal/06-05-01. Technical report (2006)