A Tool for Enterprise Architecture Analysis

Pontus Johnson, Erik Johansson, Teodor Sommestad, Johan Ullberg
Department of Industrial Information and Control Systems
Royal Institute of Technology (KTH)
{pj101, erikj, teodors, johanu}@ics.kth.se

Abstract

The discipline of enterprise architecture advocates the use of models to support decision-making on enterprise-wide information system issues. In order to provide such support, enterprise architecture models should be amenable to analyses of various properties, as e.g. the availability, performance, interoperability, modifiability, and information security of the modeled enterprise information systems. This paper presents a software tool for such analyses. The tool guides the user in the generation of enterprise architecture models and subjects these models to analyses resulting in quantitative measures of the chosen quality attribute. The paper describes and exemplifies both the architecture and the usage of the tool.

1. Introduction

During the last decade, enterprise architecture has grown into an established approach for holistic management of information systems in an organization. Enterprise architecture is model-based, in the sense that diagrammatic descriptions of the systems and their environment constitute the core of the approach. A number of enterprise architecture initiatives have been proposed, such as The Open Group Architecture Framework (TOGAF) [27], Enterprise Architecture Planning (EAP) [24], the Zachman Framework [29], Intelligrid [7], Federal Enterprise Architecture (FEA) [19] and the military architectural frameworks such as DoDAF [1], MODAF [16] and NAF [17]. What constitutes a "good" enterprise architecture model has thus far not been clearly defined. The reason is that the "goodness" of a model is not an inherent property, but contingent on the purpose the model is intended to fill, i.e. what kind of analyses it will be subjected to. For instance, if one seeks to employ an enterprise architecture model for evaluating the performance of an information system, the information required from the model differs radically from the case when the model is used to evaluate system interoperability.

Enterprise architecture analysis is the application of property assessment criteria on enterprise architecture models. For instance, one investigated property might be the information security of a system and a criterion for assessment of this property might be "If the architectural model of the enterprise features an intrusion detection system, then this yields a higher level of information security than if there is no such system." Criteria and properties such as these may be extracted from academic literature or from empirical measurements.

This paper presents a software tool for the analysis of enterprise architecture models. The tool guides the creation of enterprise information system scenarios in the form of enterprise architecture models and generates quantitative assessments of the scenarios as they evolve. Assessments can be of various quality attributes, such as information security, interoperability, maintainability, performance, availability, usability, functional suitability, and accuracy.

A number of enterprise architecture software tools are available on the market, including Metis [28], System Architect [26], Aris [23], and Qualiware [21]. Although some of these provide possibilities to sum costs or propagate the strategic value of a set of modeled objects, none have significant capabilities for system quality analysis. Within the software architecture community, various analysis methods and tools do however exist, including the Architecture Tradeoff Analysis Method (ATAM) [2], Abd-Allah and Gacek [6], Wright [1] and the Chiron-2 Software Architecture Description and Evolution Language (C2SADEL) [15]. None of these are, however, applicable in the enterprise architecture domain.

The outline of this paper is as follows. Section 2 describes the method for enterprise architecture analysis which the presented tool supports. This section also introduces an example that is elaborated throughout the paper. Section 3 briefly describes the architecture of the tool. Sections 4 to 8 go through the central components of the architecture in greater detail. The paper is concluded with a discussion and summary in Sections 9 and 10.

1541-7719/07 $25.00 © 2007 IEEE
DOI 10.1109/EDOC.2007.25

2. Method for enterprise architecture analysis

The purpose of enterprise architecture models and conducting analyses on these is to facilitate the making of rational decisions about information systems. The process of enterprise architecture analysis is illustrated in Figure 1 below. In the first step, *assessment scoping*, the problem is described in terms of a set of potential future *scenarios* of the enterprise information system and in terms of the assessment criteria (in the figure, *abstract model*) with which the scenarios will be evaluated. In the second step, the scenarios are detailed by a process of *evidence collection*, resulting in a set of models (*concrete models,* in the figure) of the different scenarios. In the final step, *analysis*, quantitative values of the models' quality attributes are *calculated*, and the models are then *visualized* in the form of enterprise architecture diagrams.

More concretely, assume that the decision maker is contemplating a change related to the customer support services in the company. The Customer Relation Management (CRM) system might be replaced by a new one, and this might also affect the system support organization. The question for the decision maker is whether the change is to the better or not.

As mentioned, in the assessment scoping step, the decision maker identifies the available decision alternatives, i.e. the enterprise information system scenarios. In this case, we find two alternatives: 1) keep the current CRM system, 2) implement a new CRM system. Also in this step, the decision maker needs to decide on how to determine which the better scenario is, i.e. the assessment criteria, or the goal of the assessment. Oftentimes, many quality attributes are desirable, high availability, high information security, high, functional suitability, high interoperability, etc. In this paper, without loss of generality, we simplify the problem to the assessment of availability of the customer support functionality.

During the next step, to identify the better alternative, the scenarios need to be detailed to facilitate an analysis of them. Much information about the involved systems and their organizational context may be required for a good understanding of their future availability. For instance, it is reasonable to believe that an experienced system administrator generally needs shorter time to repair a system than an inexperienced one. The experience level of the system administrators is thus one factor that can affect the availability and should therefore be recorded in the scenario model. The decision maker thus needs to understand what information to gather, and also ensure that this information is indeed collected and modeled.

When the decision alternatives are detailed, they can be analyzed with respect to the desirable property or properties. The pros and cons of the scenarios then need to be traded against each other in order determine which alternative is to be preferred.

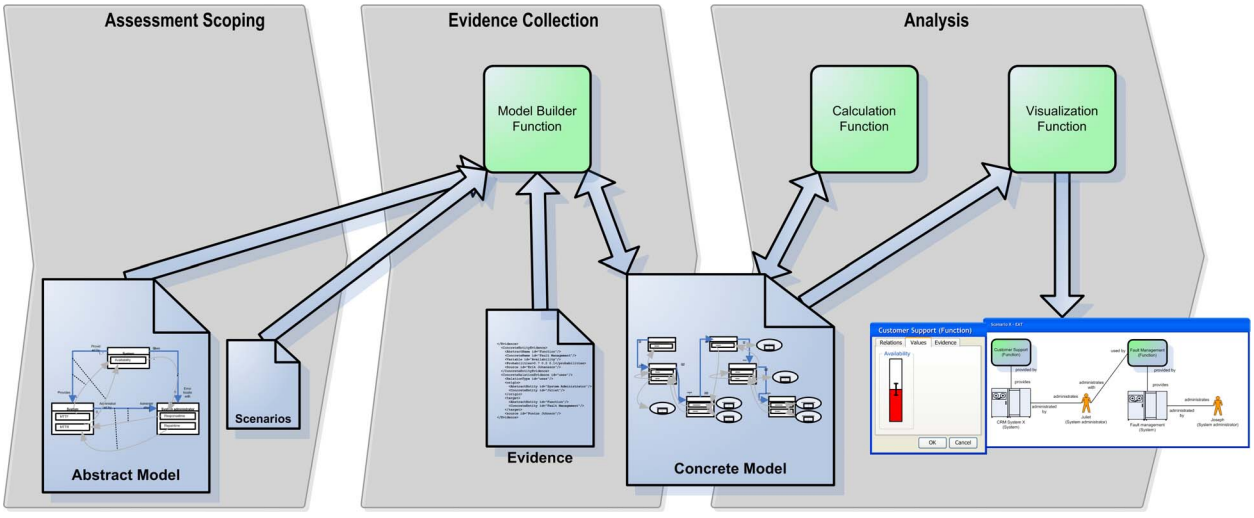

Figure 1. Illustration of the overall process of enterprise architecture analysis.

3. Design of the tool

Here we outline a software tool supporting the enterprise architecture analysis process described in the previous section. The structural design of the tool is based upon three processes, see Figure 1. Relating to the previous section, the first process step concerns the identification of the decision alternatives and the goals (assessment criteria). An example of a goal is to maximize the availability of the customer support functionality. Goals are codified in an `Abstract Model`, which also represents how various entities and attributes affect the goals. For instance, the abstract model might state that the experience of the system administrator will affect the availability of the system. Section 4 details the syntax and semantics of abstract models. The `Scenarios` are not detailed in this step, but only given a name.

In the second process step, the scenarios are elaborated. An example of a scenario might be to choose vendor X for the CRM system. In order to assess whether this is a better scenario than the one based on vendor Y, more information is required. For instance, we might need to know if the system administrator has experience of vendor X's system, since this factor will affect the goodness of the scenario. Collected information on matters like these is called `Evidence`, and the process of evidence collection is supported by the tool. Section 5 elaborates further on this topic.

From the evidence, the `Model Builder Function` creates a `Concrete Model`, which is a comprehensive enterprise architecture scenario model. The concrete model thus typically contains instantiations of entities such as `information system`, `business process`, `function`, `data`, etc. Many times, the collected evidence will not be sufficient to allow full certainty of the values of entity attributes (for instance the attribute `experience` of the entity `system administrator`). They are therefore defined as random variables, allowing the representation of uncertainty. In Section 6, the concrete model and the representation of evidence is further described.

Recall that an important purpose of enterprise architecture models is to answer relevant questions, such as whether the availability of a certain function is higher in scenario A than in scenario B. Oftentimes, it is not possible to directly collect information about properties such as availability, but it is possible to collect evidence pointing in a certain direction (e.g. whether the system administrator is experienced or not). Using Bayesian theory, the `Calculation Function` calculates the values of those attributes that could not be credibly determined directly. The tool employs the collected evidence as well as the causal relationships specified in the abstract model as input to these calculations. In Section 7, this issue is further considered.

Central artifacts in enterprise architecture frameworks and tools are graphical models. The concrete models containing the enterprise architecture scenarios therefore need to be converted into a visual format suitable for human consumption. This is performed by the `Visualization Function`, which is further detailed in Section 8.

4. Abstract model

As stated earlier, the purpose of enterprise architecture analysis is to facilitate rational IT decision making by predicting the effects of decisions on desirable system qualities, such as information security or modifiability. Generally, the causal relationships from the decisions to the qualities are complicated and in order to understand these chains the quality attributes need to be connected to causally significant intermediate concepts that are easier to relate to the decisions [11]. As an example, the experience of the system administrator may affect the availability of the system he or she administrates. The system's availability can in turn affect the availability of the provided functions, which is the quality attribute of interest. This understanding combined with knowledge about the experience of the system administrator allows predictions about the availability of the function, even though this attribute has not been measured.

It would be convenient to design enterprise architecture models so they contain the information required to perform these causal predictions. This section therefore introduces the concept of an *abstract model* which is an enterprise architecture metamodel augmented with the causal links described above.

The remainder of this section presents the different elements of abstract models. From the tool's point of view the abstract model is specified using XML, but it is also possible to depict it graphically which will be the case in this paper.

An abstract model is composed of four components: *entities, attributes, entity relationships* and *attribute relationships*. Of these, the three first entities are recognized from the class diagram notation of the Unified Modeling Language (UML) [16]. *Entities* are central in all enterprise architecture modeling and describe items of interest to the model. Sometimes these items are physical artifacts of the

real world, such as "person" or "computer", and sometimes they are more abstract like "process", "function" or "data". In the abstract model, the entities are depicted essentially the same way as classes in UML, a box with the name of the entity at the top of the box and a line separating the name area from the rest of the box.

The entities can be connected to each other with *entity relationships*. These relationships are represented as lines between the entities and, in both ends of the relationship, role names (e.g. "uses" and "used by") are used in accordance with the UML. The multiplicity is indicated on the entity relationship.

Attributes are contained by entities, but in contrast to the UML, they are random variables (thus paving the way for probabilistic inference). The attributes may assume values from a finite domain such as {True, False}, {High, Medium, Low}, {1-10, 11-100, 101-1000}, etc. They can represent concepts such as "Age", "Experience", "Usability", or "Interoperability". For instance *P(*`Experience=High`*)* represents the probability that the attribute `Experience` will assume the value `High`. Attributes are portrayed as rectangles within the entities. The range of possible values of an attribute is defined in the abstract model, but this is not illustrated in the graphical notation shown in Figure 2.

The last component of the abstract model is the attribute relationship. Graphically, they are illustrated as gray arrows between the rectangles denoting attributes and the relationship indicates a probabilistic dependence between two attributes. If this attribute relationship spans two entities, it is always associated with a particular entity relationship, which is denoted by the line connecting the attribute relationship with the entity relationship. As an example, the attribute `Experience` of the entity `System Administrator` will affect the attribute `Availability` of the entity `System` only if the entity `System Administrator` `Administers` this particular `System`, see Figure 2.

When there is a relationship between two attributes, a change in the state of the source (i.e. the parent) attribute is expected to lead to a change in the target (child) attribute. The probability of a target attribute assuming a certain state thus depends on the values of its parent nodes. This probabilistic dependence is specified fully by a *conditional probability distribution*, $P(X=x \mid Y=y, Z=z)$, stating the probability that a target attribute X will assume a certain value x, given that its parents Y and Z have assumed some values y and z.

These conditional probabilities are used when performing the assessment of quality attributes (see Section 7 for more information) and are expressed in *conditional probability tables*, where each possible configuration of the parent attributes' states is related to a probability distribution over the child attributes domain.

There are basically three sources of information when creating abstract models. Firstly, it is possible to interpret scientific literature on the chosen topic (e.g. books and articles on the assessment of the quality attribute availability). Secondly, it is possible to elicit such knowledge from experts [1]. Thirdly, empirical data may be employed to parameterize an abstract model. Methods have also been developed to base conditional probabilities on combinations of sources [5].

However, the size of a conditional probability table grows exponentially with the number of influencing attributes. For large models, it is therefore a significant task to determine these tables by specifying each cell individually. One approach to counter this problem is to express generalized conditional probability tables by parameterized functions such as *noisy-max*, *noisy-min* [1] [25]. The number of parameters in these models typically grows linearly instead of exponentially with the number of parents. Furthermore, as will be exemplified in Section 6, these parameterized functions are required for the efficient use of multiplicity.

As stated earlier, this paper will present an example where the goal is to assess the availability of the customer support function provided by a customer relation management system in an enterprise. The example is, of course, simplified for pedagogical reasons and would be extended significantly if an actual assessment were to be performed.

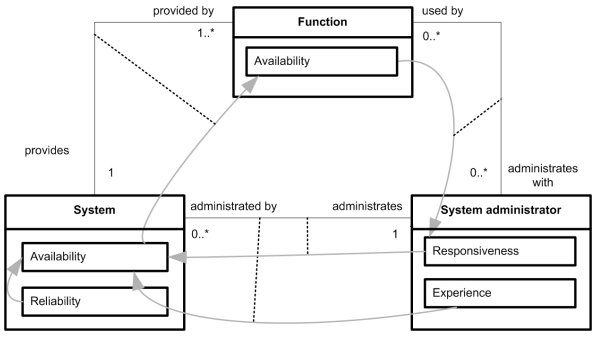

Figure 2. Illustration of the abstract model for the example presented in this paper.

The abstract model used in this example codifies a theory of the availability of functions. The theory starts by the claims that the availability of a function is dependent on the availability of the system that provides the function, see Figure 2. The system's availability, in turn, depends on its reliability, i.e. how often the system fails. The system availability is also dependent on the time needed for the person administrating the system to identify and prepare for correcting the failures (his or her responsiveness). Furthermore, when the administrator has arrived to repair the system, his or her experience will affect the repair time. Hence experience of the system administrator is of relevance. Finally, if the system administrator is supported by any functions in their administrative work, such as fault management functionality, the availability of these will have an impact on the responsiveness of this administrator.

The model's conditional probability tables are expressed by parameterized functions. A min-function describes how the systems availability is influenced by its parents. This function implies that the availability of a system is equal to the minimum of its administrator's qualities and its own reliability. The availability of a function is modeled as identical to the availability of the system providing this function. Finally, the dependence of the responsiveness of the system administrator on the availability of functional support is modeled as a max-function.

5. Evidence collection

Given the abstract model, detailed information needs to be gathered about the scenarios so that the assessment can be performed. This is the evidence collection process. The process of information gathering is guided by the tool so that only evidence relevant to the assessment is collected. The evidence is used to create one concrete model per scenario, cf. Section 6, where the generic concepts of the abstract model are instantiated as enterprise-specific (concrete) entities and attributes.

Recall that before evidence collection starts, assessment scoping includes defining the goal of the assessment and identifying the scenarios to choose between. In our example the scenarios are 1) to keep the existing CRM system or 2) to purchase and install a new one. The goal in the example is limited to maximizing the availability of a function.

When the scope of the assessment has been determined, evidence is collected. This is typically done by reading documents, performing interviews or from first-hand experience. Three types of evidence can be supplied, evidence on the existence of various entities, evidence on the relationships between entities, and evidence on the value of the attributes. The first two types determine the structure of the concrete model and the last type fills the structure with indications of attribute states so that the quality assessment can be performed. Only evidence on entities, relationships and attributes present in the abstract model is permissible evidence.

During evidence collection, contextual information about the evidence is also gathered, for instance how old it is and from what source it was elicited. This information enables an estimate of the evidence's credibility. [10] The evidential credibility is subsequently used to estimate the credibility of the assessment as a whole.

Returning to the example, the scoping declares that there is a Function called Customer Support. Based on this, the tool guides the user to supply the evidence needed to perform the assessment. The abstract model states that all functions are provided by systems; in this example the customer support function is provided by a CRM system. Systems do, according to the multiplicity constraints in the abstract model, have exactly one system

administrator. Hence, evidence should be provided on who the system administrator of the system is in the two scenarios. Furthermore, system administrators could use functions in their administrative work. If they do so, evidence is collected on these functions and in accordance with the abstract model, evidence is collected on entities related to these functions, identifying the systems providing these functions and so on.

Examples of evidence regarding a system administrator are shown in Figure 3. The first part of the file is a piece of evidence on entities specifying that John Smith claims that there is a system administrator called Juliet, i.e. that there exists an instantiation of the entity `System Administrator` of the abstract model with the name `Juliet`.

```
<Evidence>
    <EntityEvidence>
        <AbstractEntityName>SystemAdministrator</AbstractEntityName>
        <ConcreteEntityName>Juliet</ConcreteEntityName>
        <Source>John Smith</Source>
        <SourceDate>2007-03-05</SourceDate>
        <SourceType>Interview</SourceType>
    </EntityEvidence>
    ...
    <VariableEvidence>
        <AbstractEntityName>SystemAdministrator</AbstractEntityName>
        <ConcreteEntityName>Juliet</ConcreteEntityName>
        <ConcreteVariableName>Experience</ConcreteVariableName>
        <Source>Juliet</Source>
        <Statement>High</Statement>
        <SourceDate>2007-03-10</SourceDate>
        <SourceType>Interview</SourceType>
    </VariableEvidence>
    ...
</Evidence>
```

Figure 3. XML-segment of evidence.

In the second part, a piece of evidence on the value of attributes is given where Juliet herself claims that her experience is high. Her answer is supplied together with contextual information about when and by which means the evidence was collected. This information will be used to calculate the credibility of her statement. In the same way other evidence relevant to the assessment is collected and specified.

6. Concrete model

At this point, we have discussed the abstract model and the collected evidence. Together, these two can be combined into a concrete model. A concrete model is an instantiation of an abstract model much in the same way that an object model is an instantiation of a class model in object-oriented modeling. Whereas the abstract model speaks of general entities, attributes and relationships, the concrete model speaks of these in the context of a specific enterprise or information system. In our example, the abstract model describes the general influence of a system administrator's experience on the availability of the administered system. The concrete model, however, specifies *which* system and *what* administrator are under consideration in this particular instance. The actual instantiation process is straightforward. If there is a piece of evidence that an entity or relationship exists or that a certain attribute has a certain value, then this is introduced into the concrete model. Considering the example in this paper, Figure 4 is the result of the combination of the abstract model in Figure 2 and a set of evidence partially described in Section 5 above.

According to the multiplicity in the abstract model, a function is always provided by exactly one system.

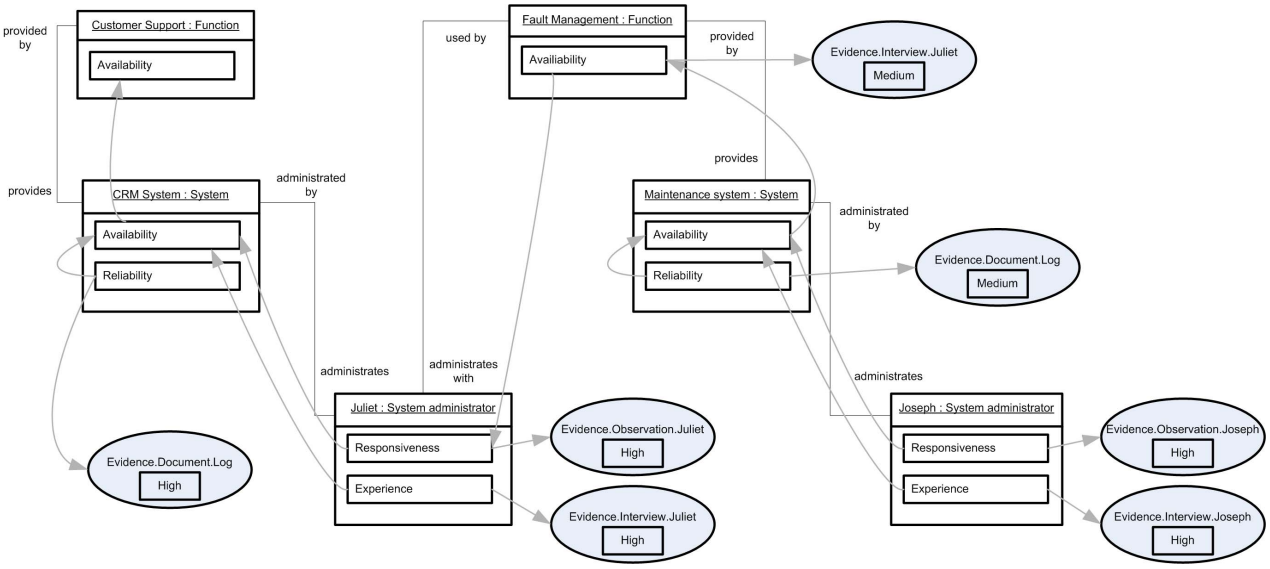

Figure 4. Illustration of a concrete model for one of the scenarios discussed in this paper.

The customer support function is in this case provided by the CRM system. The availability of the customer support function is, as expressed in the abstract model, dependent on the availability of the system it is provided by, producing the arrow between the two attributes. The abstract model further stipulates that every system has exactly one system administrator; in the case of the CRM system, that administrator is Juliet and her responsiveness and experience has an impact on the CRM system's availability.

Juliet uses a fault management function to detect failures in the systems she administers. Since she will not receive failure notifications when the fault management function is unavailable, her responsiveness depends on the availability of that function. The fault management function is in turn provided by a maintenance system which is administrated by another system administrator, Joseph. A system administrator could use any number of functions as support in the administrative work. Unlike Juliet however, Joseph does not use any administration-supporting functionality. His responsiveness and experience do, however, still impact the availability of the fault management system that he administrates.

Given the relationships between entities, the abstract model provides information on how attributes influence each other. This influence is expressed in a conditional probability table where states of influencing attributes determine the probability distribution over the states in the target attribute. In this case, the abstract model details that the availability is a min-function of its influencing attributes (cf. Section 4). The conditional probability table corresponding to this function is described in Table 1.

Table 1. Conditional probability table for the availability of the entity CRM system.

CRM_System.Reliability		High									Medium									Low								
Juliet.Responsiveness		High			Medium			Low			High			Medium			Low			High			Medium			Low		
Juliet.Experience		H	M	L	H	M	L	H	M	L	H	M	L	H	M	L	H	M	L	H	M	L	H	M	L	H	M	L
CRM_System.Availability	H	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	M	0	1	0	1	1	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	L	0	0	1	0	0	1	1	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

In the example elaborated here, evidence has been collected on the state of certain attributes, but not all. As shown by ellipses in Figure 4, there is evidence on the two administrators' attributes and the two system's reliability. Furthermore, there is evidence on the availability of the fault management function.

The experience of the system administrator is one of three attributes which according to the model

influences the CRM system's availability. As indicated in Figure 3, Juliet was asked about her experience during an interview in the evidence collection process, and she replied that her experience is high.

There is, however, an inherent uncertainty regarding whether Juliet's answer corresponds to her actual experience. This uncertainty may be estimated in the light of the contextual information gathered about the source and age of the evidence. The relationship between Juliet's response and her actual experience is expressed in Table 2.

Table 2. Conditional Probability table for evidence on Juliet's experience.

Juliet.Experience		High	Medium	Low
Evidence.Interview.Juliet	High	0.8	0.1	0.1
	Medium	0.1	0.8	0.1
	Low	0.1	0.1	0.8

To each piece of evidence supplied about the state of attributes, a conditional probability table like the one in Table 2 is created.

Based on the design of the abstract model, the concrete model now holds all the information necessary to assess the availability of the customer support function. It contains the entities, relationships and attributes which directly or indirectly have relevance to the assessment, together with evidence on the states of the attributes. In a decision situation, it would be reasonable to create one or a few similar scenarios to allow comparison.

7. Calculation

In the previous section, the combination of abstract model and collected evidence resulted in a concrete model consisting of entities, relationships and entity attributes with assigned values. There were, however, still attributes with unassigned values. For instance, in the running example, the availability of the CRM system is still unknown.

The attribute relationships from the abstract model allow us to calculate these values. Recall that an attribute relationship represents a probabilistic dependence of the target attribute on the value of the source attribute and that these dependencies may be represented in conditional probability tables. In Table 1, we presented the conditional probability table of the `CRM system` attribute `Availability`, dependent on the system's `Reliability` and the administrator's `Experience` and `Responsiveness`. If the probability distributions of the source attributes are known, it is possible to infer a value for the target attribute using the law of total probability,

$$P(A) = \sum_i P(A|B_i)P(B_i)\cdot$$

If, for instance, we know the values of the system's reliability, $RL=rl_j$, and the administrator's responsiveness, $RS=rs_l$, then we can calculate the probability that the availability assumes a certain value, $AV=av_i$, using the following formula,

$$P(AV = av_i \,|\, RL = rl_j, RS = rs_l) = \sum_{k\in\{H,M,L\}} P(AV = av_i \,|\, RL = rl_j, EX = ex_k, RS = rs_l)\cdot P(RL = rl_j)P(EX = ex_k)P(RS = rs_l)$$

where $P(AV=av_i|RL=rl_j,EX=ex_k,RS=rs_l)$ is given by Table 1.

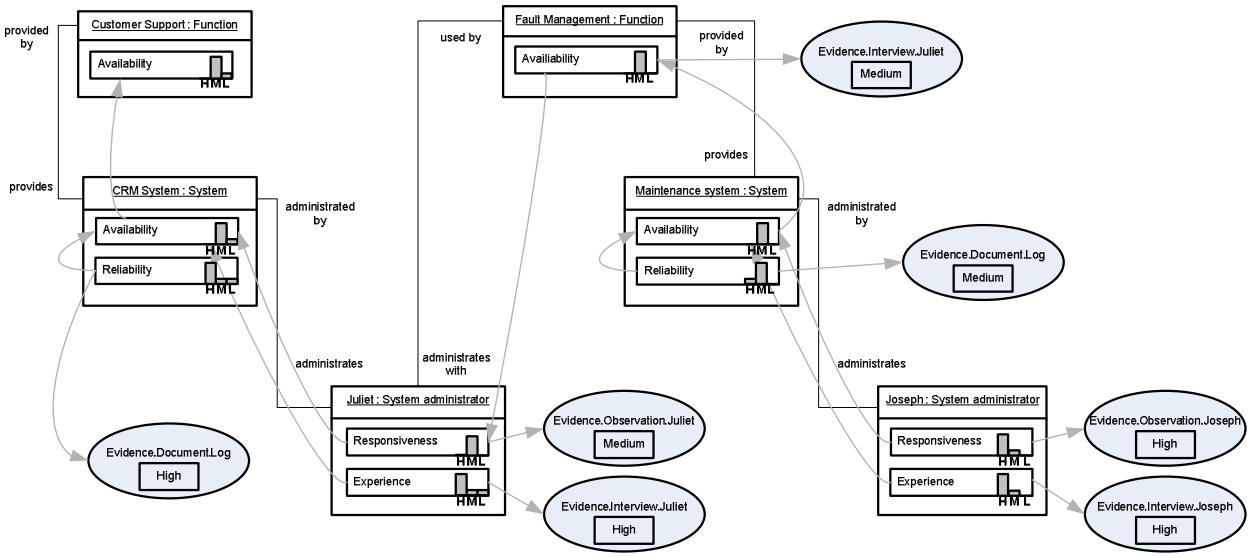

Figure 5 Attribute values in the concrete model are inferred from the collected evidence using the probabilistic dependencies between attributes (grey arrows). The values are presented diagrammatically as probability distributions over the values *High (H), Medium (M)* and *Low (L)*.

Using Bayes' rule,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},$$

the probability distributions of the nodes *RL, EX* and *RS* may be calculated from evidence collected on their state. Table 2 in Section 6 represents our confidence in Juliet's statement, JEX, regarding her own experience, $P(JEX=jex_i|EX=ex_j)$. Given that Juliet provides an answer, JEX=*jex*, our belief of Juliet's experience becomes

$$P(EX = ex_j | JEX = jex) = \frac{P(JEX = jex | EX = ex_j) P(EX = ex_j)}{P(JEX = jex)}$$

where $P(JEX=jex)$ is a normalization term. Bayes' rule also allows the pooling of multiple pieces of evidence, so that, e.g., also Joseph's opinion on Juliet's experience can influence our beliefs.

It is possible to obtain probability distributions for attributes also when only one or a few surrounding nodes are known, although the resulting beliefs normally are associated with a lower certainty. Figure 5 presents the concrete model of the running example, with a set of evidence of the same credibility as Juliet's statement (Table 2). The relationship between system availability and function availability are described in Table 3 below and the relationship between the fault management function's availability and Juliet's responsiveness is shown in table 4.

Table 3. Conditional probability table for the CRM system's availability.

CRM_System.Availability		High	Medium	Low
Customer_Support.Availability	High	1	0	0
	Medium	0	1	0
	Low	0	0	1

Table 4. Conditional probability table for Juliet's responsiveness.

Fault_Management.Availability		High	Medium	Low
Juliet.Responsiveness	High	1	0	0
	Medium	0	1	0
	Low	0	0	1

As can be seen from the figure, in this example a limited set of evidence is sufficient to produce interesting results. In particular, these results can be used to support decisions between alternative future scenarios. If a higher confidence is desired in order to comfortably make a decision, it is necessary to engage in further evidence collection.

8. Visualization

When the evidence has been collected and the

calculation has been performed, the results need to be visualized to the decision maker. The results are comprised of the concrete model with inferred values, and are thus conveniently presented in the form of a traditional enterprise architecture model.

Figure 6 presents two scenario models. The first model is from the running example, while the second one is an alternative where the system administrator Juliet is additionally supported by an intrusion detection system. The functions are depicted as boxes, the systems as mainframes and the system administrators are represented as persons.

By clicking on the entities, a dialog box presents the attributes and their values. In the figure such a property window is shown for the function `Customer Support` where the inferred availability of the function is displayed. The credibility of the assessment is presented in the form of an I-bar.

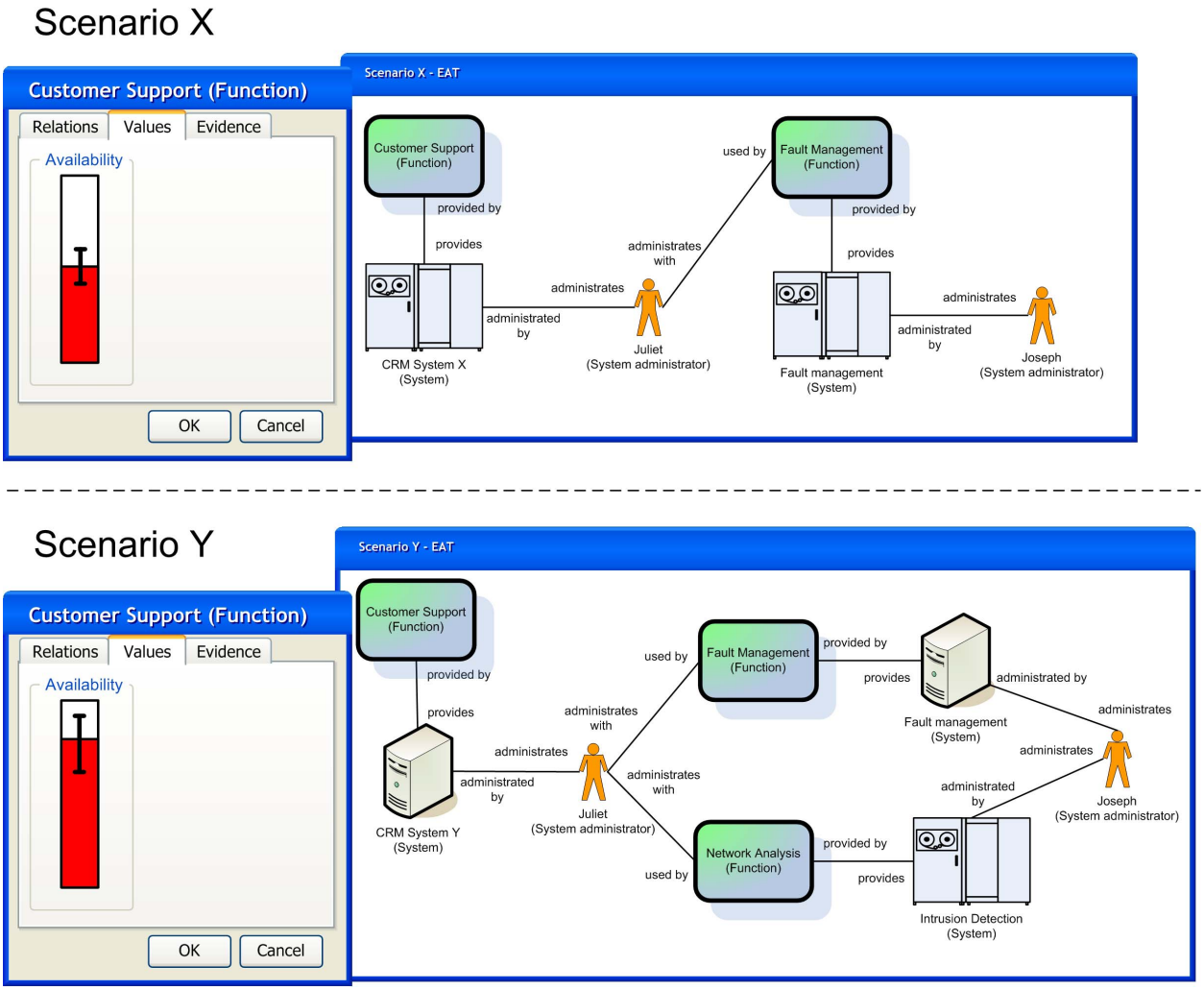

Figure 6. Illustration of visualized results for the scenarios discussed in this paper.

By considering the two scenarios and their calculated qualities, a decision maker is in a good position to make a rational decision with respect to the evolution of the enterprise information system.

9. Discussion

At the time of writing, we have developed a prototype for the core functions, reading evidence and abstract model and from these creating a concrete model with the values of the attributes calculated.

Apart from validating the current prototype further there are two major research questions that will be focused upon within the nearest future, first there is the issue of the exponential growth of the conditional probability matrices, some approaches to tackle this have already been mentioned and others that are being discussed are the introduction of intermediary attributes [18] or removing arcs where the influence is weak [13]. The second question is how to best select which evidence to collect, a topic that becomes even more important as the models grows larger [9].

10. Summary

In this paper, we have presented a tool for analysis of enterprise architecture scenarios. The tool guides the development of enterprise architecture models and from these derives a measure of the quality of the modeled architecture. In the paper, the exemplified quality measure is the availability of a certain information system function, but the tool supports the analysis of various quality attributes, such as information security, interoperability, maintainability, performance, and more.

Information system decision making is supported by allowing quantitative comparisons of the qualities of possible future scenarios of the enterprise information system and its context. The tool also provides an estimate of the credibility of the quantitative assessment.

11. Acknowledgements

The authors would like to thank the sponsor of this research, Elforsk AB (owned jointly by Swedenergy and The Swedish National Grid).

12. References

[1] Allen, R., R. Douence, D. Garlan, "Specifying and Analyzing Dynamic Software Architectures" *Proceedings of the 1998 Conference on Fundamental Approaches to Software Engineering*, 1998

[2] Clements, P., R. Kazman, M. Klein, "*Evaluating Software Architectures: Methods and Case Studies*", Addison-Wesley, 2001

[3] DoD Architecture Framework Working Group, "*DoD Architecture Framework*" Version 1.0, 2003

[4] Druzdzel MJ and L.C. van der Gaag, "Building probabilistic networks: where do the numbers come from?" *IEEE Trans Knowledge Data Eng* 2000;12:481–6.

[5] Druzdzel, M.J. and L.C. van der Gaag, "Elicitation of Probabilities for Belief Networks: Combining Qualitative and Quantitative Information," *Proc. 11th Conf. Uncertainty in Artificial Intelligence*, pp. 141-148, 1995

[6] Gacek, C., "*Detecting Architectural Mismatch During System Composition*", PhD. Thesis, University of Southern California, 1998

[7] Hughes, J., *The Integrated Energy and Communication Systems Architecture*, Electric Power Research Institute, 2004

[8] Jensen, F., "*Bayesian Networks and Decision Graphs*", Springer-Verlag, 2001

[9] Johansson, E., M. Ekstedt, P. Johnson, "Assessment of Enterprise Information Security – The Importance of Information Search Cost", *Proceedings of the 39wh IEEE Hawaii International Conference on System Sciences*, Hawaii, 2006

[10] Johansson, E. and P. Johnson, "Assessment of Enterprise Information Security – Estimating the Credibility of the Results", *Proceedings of the 9th IEEE international Annual Enterprise Distributed Object Computing Conference*, The Netherlands, 2005.

[11] Johnson, P. and M. Ekstedt, *Enterprise Architecture – Models and Analyses for Information Systems Decision Making*, Studentlitteratur, Sweden, 2007

[12] Johnson, P. et. al. "Enterprise Architecture Analysis with Extended Influence Diagrams", *Information Systems Frontiers*, Volume 9, issue 2-3, 2007

[13] Kjaerulff, U., "Reduction of Computational Complexity in Bayesian Networks through Removal of Weak Dependences," *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pp. 374-382, 1994

[14] Lagerström, R., P. Johnson, P. Närman, "Extended Influence Diagram Generation", *Proceedings of the third International Conference on Interoperability for Enterprise Software and Applications,* Springer-Verlag Portugal, 2007

[15] Medvidovic, N., D. Rosenblum, R. Taylor, "A Language and Environment for Architecture-Based Software Development and Evolution", *Proceedings of the 21st International Conference on Software Engineering*, 1999

[16] MoDAF, "*Ministry of Defense Architecture Framework*", Version 1.0, 2005

[17] NAF, "*NATO C3 Technical Architecture*", Volume 1-5, Version 7.0, Allied Data Publication 34, 2005

[18] Olesen, K.G. et. al. "A MUNIN Network for the Median NerveÐA Case Study on Loops", *Applied Artificial Intelligence*, vol. 3, pp. 385-404, 1989

[19] OMB – U.S. Office of Management and Budget, "*FEA Consolidated Reference Model Document*", Version 2.1, 2006

[20] OMG, *Unified Modeling Language: Superstructure*, version 2.1.1, February 2007

[21] Qualiware, *Qualiware*, http://www.qualiwareinc.com/, accessed May 2007

[22] Pearl, J., "*Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*", Morgan Kaufmann, 1988.

[23] Scheer, A.-W., *Business Process Engineering: Reference Models for Industrial Enterprises*, 2nd ed., Springer-Verlag, 1994.

[24] Spewak, S.H., Hill. S.C., "*Enterprise Architecture Planning. Developing a Blueprint for Data, Applications and Technology*", John Wiley and Sons, 1992

[25] Srinivas S., "A Generalization of the Noisy-OR Model", *Proceedings of 9th Conference on Uncertainty in Artificial Intelligence,* pp. 208-215, 1993.

[26] Telelogic, *System Architect*, http://www.telelogic.com/products/systemarchitect/, accessed: May 2007.

[27] The Open Group. *The Open Group Architecture Framework*, Version 8, 2005

[28] Troux Technologies, *Metis*, http://www.troux.com/products/, accessed May 2007

[29] Zachman, J., *A framework for information systems architecture,* IBM Systems Journal, 26(3), 1987