Using Enterprise Architecture Models for System Quality Analysis

Per Närman, Marten Schönherr, Pontus Johnson, Mathias Ekstedt, Moustafa Chenine
Department of Industrial Information and Control Systems,
Royal Institute of Technology (KTH), Technische Universität Berlin (TUB)
{pern, pj101, mek101, moustafac}@ics.kth.se; {marten.schoenherr}@telekom.de

Abstract

Enterprise Architecture is a model-based approach to business-oriented IT management. To promote good IT decision making, an Enterprise Architecture framework needs to explicate what kind of analyses it supports. Since creating Enterprise Architecture models is expensive and without intrinsic value, it is desirable to only create Enterprise Architecture models based on metamodels that support well-defined analyses.

This paper presents the content and extension of a metamodel which supports creating models containing the information necessary to conduct system quality analyses, specifically with respect to availability, accuracy, confidentiality and integrity. The metamodel is an extension and formalization of the metamodel underlying the ArchiMate modelling language for Enterprise Architecture. The use of the extended metamodel is demonstrated in a case study where the availability, accuracy, confidentiality and integrity of the two Service Oriented Architecture (SOA) platforms Sun JCaps and PrOSeRO were evaluated.

1. Introduction

In recent years, software, system and Enterprise Architecture (EA) have become established disciplines in both industry and academia. Architecture models may aid the communication between various stakeholders and facilitate comprehension of the properties of the complex systems they represent. One important part of this comprehension is to be able to infer new knowledge from a model, i.e. to be able to analyze the models.

Looking at common EA frameworks, however, analysis is seldom explicitly addressed. Although modeling and metamodeling is an important part of the frameworks and they commonly use views and viewpoints as a way of introducing the reasons and purposes of EA models, it is hardly ever explicated how the analyses should be carried out from any specific viewpoint. For instance, it might be recognized that an information security viewpoint is needed, but how to actually conduct the information security analysis is left to the user. Exactly how do all the model entities of the viewpoint influence each other, and how is the concept of information security depicted in the model? In most cases it is not even evident exactly what entities that should be included in the viewpoint in the first place. However, by introducing explicit inference rules for these things it is possible to perform the analysis. There are architecture analysis methods, such as the Architecture Tradeoff Analysis Method (ATAM), [1] but these are general and do not postulate precisely what must be captured in the architecture models.

The present work builds on previous research within the field of architecture analysis where architecture models are analyzed using a formalism based on Bayesian statistics [2][3][4]. This approach allows the analysis of various system properties, such as the interoperability, information security and the availability of software systems.

This paper describes how this Bayesian analysis approach is used for estimating availability, accuracy, confidentiality and integrity of two SOA platforms. The purpose of the case study is to provide decision-makers with hard facts concerning the Quality of Service offered to the business when using either one of the two SOA platforms to orchestrate services for a particular business process. Even though application Quality of Service may very well be affected by for instance the quality of the IT organization, we delimit the scope of our evaluation to only concern the applications themselves and their supporting infrastructure.

The next section provides some background to the architecture analysis approach on which this paper is based. Section 3 gives a short overview of a number of EA frameworks and a review of their suitability for being complemented with analysis inference rules. In
section 4 the original ArchiMate modeling language is presented. Section 5 proceeds to detail the extensions and adaptations of ArchiMate to make it suitable to the system property analyses mentioned above and Section 6 describes a case study in which the extended ArchiMate metamodel is applied for analyzing the two SOA platforms. Section 7 concludes the paper.

2. Abstract and concrete models

An abstract model is an EA metamodel containing entities and entity relations, augmented with attributes and attribute causal relations in representing the nodes and relations of a Bayesian network [6][7]. With “metamodel” we mean a class of models belonging to the M1 level of OMG’s four-layered architecture [5].

Entities are fundamental parts found in most metamodels. Entities represent the objects of interest when modeling, e.g. application, services, persons, or processes. Entities in abstract models are similar to classes found in UML [5].

Entity relations connect two entities, e.g. “Interface is provided by Application” or “Person is a resource of a Process”. Entity relations also state the multiplicity of the relationship between the entities, e.g. that one person can be the resource of zero or more processes.

Attributes of an abstract model represent variables related to the entities. UML also have attributes related to entities, but the attributes in abstract models differ from the attributes in UML. In abstract models, attributes and attribute relations represent the nodes and relations of a Bayesian network, see below. A richer account of abstract and concrete models is found in Johnson et al. [6].

2.1. Bayesian networks

Friedman et al. [8] describes a Bayesian network, \( B=(G, P) \), as a representation of a joint probability distribution, where \( G=(V, E) \) is a directed acyclic graph consisting of vertices, \( V \), and edges, \( E \). The vertices denote a domain of random variables \( X_1, \ldots, X_n \), also called chance nodes. In the context of abstract models, each chance node corresponds to an attribute. Each chance node, \( X_i \), may assume a value \( x_i \) from the finite domain \( \text{Val}(X) \). The edges denote causal dependencies between the nodes, i.e. the causal relations between the nodes. The second component, \( P \), of the network \( B \), describes a conditional probability distribution for each chance node, \( P(X_i) \), given its parents \( \text{Pa}(X_i) \) in \( G \). It is possible to write the joint probability distribution of the domain \( X_1, \ldots, X_n \) using the chain rule of probability, in the product form

\[
P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Pa}(X_i))
\]

In order to specify the joint distribution, the respective conditional probabilities that appear in the product form must be defined. The second component \( P \) describes distributions for each possible value \( x_i \) of \( X_i \) and \( \text{pa}(X_i) \) of \( \text{Pa}(X_i) \), where \( \text{pa}(X_i) \) is the set of values of \( \text{Pa}(x_i) \). These conditional probabilities are represented in matrices, here forth called Conditional Probability Matrices (CPMs). Using a Bayesian network, it is possible to answer questions such as what is the probability of variable \( X \) being in state \( x_i \) given that \( Y = y_2 \) and \( Z = z_1 \).

In the general case, the relations between variables described by the conditionally probability matrices can be arbitrarily complicated conditional probabilities.

![Figure 1: A Bayesian Network, an abstract model and a concrete model for availability analysis.](image)

The models described further ahead in this paper use the simple AND relation. The CPM for the AND relation used in the availability example of Figure 1 is shown in Table 1.

<table>
<thead>
<tr>
<th>Availability of O/S</th>
<th>Up</th>
<th>Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability of HW</td>
<td>Up</td>
<td>Down</td>
</tr>
<tr>
<td>Availability of App.</td>
<td>Up</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Down</td>
<td>1</td>
</tr>
</tbody>
</table>

The real world rarely allows itself to be described by AND relations and to be able to represent uncertainty we insert probabilistic data as prior evidences in chance nodes with no parents and as new attributes in other chance nodes. We are thus able to say that there is a 95% chance that a system, for instance, is in the state ‘Up’ and 5% chance that it is in the state ‘Down’.

More comprehensive treatments on Bayesian networks can be found in e.g. Neapolitan [9], Jensen [10], Shachter [11] and Pearl [12].
2.2. Creating concrete models

The abstract model tells us what information we need to find in order to conduct analyses of different variables. Once this information is collected it is specified in the model, thus creating an instantiation of the abstract model. These instantiations are called concrete models and correspond to the M0 level of the OMG architecture.

Once a concrete model has been created we can use the Bayesian mathematics to calculate the values of the attributes of the model.

3. Enterprise Architecture Metamodels

Currently there exist a number of EA frameworks, approaches, and tools. Some of these are more model-oriented than others. In this work we are interested in finding an EA metamodel that includes entities and relationships that can be used for system quality analysis.

Out of the many EA frameworks we examined, four frameworks where chosen for to be evaluated in greater detail to see how suitable they were for the purpose of system quality analysis. The examined frameworks were: DoDAF [13], ArchiMate [14], Zachman [15], and the metamodel from the EA tool Troux Architect [16].

After the comparison it was concluded that ArchiMate’s metamodel was well tailored to perform system quality analysis, with such analysis’ focus on system architecture issues.

Specifically ArchiMate was outstanding with respect to resolution; its metamodel treats the infrastructure, application and business layer on a lower level of abstraction than does for instance DoDAF. Furthermore, ArchiMate is not designed with any specific industry in mind as in the case of DoDAF, nor is it proprietary as is the Troux Architect metamodel.

4. ArchiMate

ArchiMate is an EA framework [14] developed by the ArchiMate Foundation consisting of primarily Dutch companies and research institutions.
The ArchiMate metamodel distinguishes between three layers; the business, application and technology layer, where the technology supports the applications and the applications support the business.

The entities of the layers are in turn categorized according to three aspects of EA; structure, information and behavior.

The service concept has been used extensively in the ArchiMate language. Services encapsulate and hide the internal behavior of underlying layers and give the overlying layers access to functionality from underlying layers through well-defined interfaces. For instance, an application service is seen to encapsulate a number of internal application functions to make them useful to the business actors of the business processes. By using services the business actors neither have to consider the internal behavior of the information systems nor the technology used to realize the behavior. Figure 2 shows the original ArchiMate metamodel [17].

5. Adapting ArchiMate

Even though the ArchiMate metamodel is highly useful when depicting an EA, it merely provides the skeleton needed for the system quality assessments envisaged in this paper. To conduct system quality analyses, ArchiMate needs to be extended into the abstract model form.

This transformation entails adding attributes to the metamodel entities, causal relations that connect the attributes to other attributes and finally the conditional probability matrices (CPMs) that mathematically express the causal dependencies between the attributes.

The causal relations between the attributes of the abstract model were deduced using the definitions of the entities and the entity relations of the ArchiMate abstract model. For instance, the Application Service entity is realized by one or more Application Functions, where an Application Function models the internal functional behavior of an information system. This definition implies that if one of the Application Functions goes down, the Application Service goes down, which translates into a causal dependence between the availability of the Application Functions and the availability of the Application Service where the former affects the latter.

In some cases the attributes of the entities are affected by phenomena not shown in the abstract model. The availability of an Application Component could for instance be 98% even though its supporting infrastructure is 100% available. To capture this we introduce the concept of “inherent” system qualities, referring to the probability of one of the attributes being in a certain state regardless the state of the entity’s surrounding environment. To preserve the AND structure of the CPMs the inherent qualities were inserted either as prior probabilities or new attributes in the GeNIe [18] model used for computations.

We now turn to the specific system qualities and how they have influenced the abstract model. The full abstract model is described in section 5.5. Since the focus of the evaluations was Quality of Service delivered to the Business layer, we have chosen not to include business-related entities or entities related to the IT organization in the abstract model.

5.1. Availability

Availability is the ratio between a system’s uptime or its Mean Time To Failure (MTTF) and the total time of operation, which is the added uptime and downtime. Downtime is often referred to as Mean Time To Repair [19]. Mathematically we therefore have

\[
\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}
\]

From an IT management as well as a user perspective, the “top-level” entity for availability assessments is the Application Service class. It is the availability of Application Services that will be defined in for instance Service Level Agreements.

The CPMs that describe availability contain the states “Up” and “Down”.

5.2. Accuracy

The accuracy of an information system is measured by the degree to which it, given correct input data, produces accurate output data [20].

Although commonly associated with Data Objects, accuracy also has to do with functionality; if the computations are erroneous, the output will be inaccurate. Regardless of what causes inaccuracy it is the Application Service class that is the top-level class in accuracy evaluations.

In the accuracy CPM the entities may assume the states “Accurate” and “Inaccurate”.

5.3. Confidentiality

According to ISO/IEC 27001:2005 standard for information security [21], confidentiality is defined as
“the property that information is not made available or disclosed to unauthorized individuals, entities, or processes”.

Based on the above, confidentiality obviously is a property that primarily concerns Data Objects and casually speaking means the ability of outsiders to “read” Data Objects. To trace what affects this confidentiality we postulate that confidentiality does not exclusively concern Data Objects but also other entities in the metamodel. In those cases “Confidentiality” means the ability to read Data Objects being processed by those metamodel entities. So for instance, if an unauthorized person is able to tap into the Network and read the network traffic, this constitutes a violation of Network confidentiality.

The confidentiality CPM has two states “Compromised” and “Not Compromised” meaning that either some unauthorized agent has gained “read” access or it has not.

5.4. Integrity

According to the ISO 27000 standard, integrity is “the property of safeguarding the accuracy and completeness of assets” [21].

As with confidentiality, integrity chiefly concerns Data Objects which is the target of the evaluation here.

Analogous with the confidentiality assessments, integrity is here taken to mean roughly the ability of unauthorized agents to “write” to Data Objects and the property is generalized to other entities of the metamodel as well. In the Network case for instance, whenever an agent is able to disturb or alter Network traffic, the integrity of the Network is compromised.

As in the confidentiality case, the states of the CPM are “Compromised” and “Not Compromised”.

5.5. The resulting abstract model

The entire abstract model for system quality assessments can be found in Figure 3. Compared with the original ArchiMate metamodel some metamodel entities have been removed, some relations are different or removed, and attributes with attribute relations have been added.

Application Services encapsulate functionality that is meaningful to the surrounding environment; the application users in business processes [17]. As mentioned, the Application Service is here the top-level attribute for availability and accuracy assessments. These attributes were therefore inserted in the Application Service entity.

Application Services have no inherent availability or accuracy, since they are abstract concepts entirely realized by Application Functions. [17]. The availability and accuracy of an Application Service is therefore contingent on its constituent Application Functions which translates to a causal connection between the availability and accuracy of the Application Functions and the Application Services. Application Functions are concrete functions with an inherent availability; regardless of the state of the surrounding environment they may stop working correctly, for instance due to bugs.

Figure 3: An abstract model for system quality analysis.

Further on, Application Functions may trigger other functions. An Application Function’s availability may therefore be dependent upon the availability of another Application Function.

The Application Functions are implemented in Application Components. An
Application Component represents an information system or part of an information system. The Application Functions implemented in a specific Application Component will not function if the entire component goes down; therefore, there is a causal relation between Application Component availability and Application Function availability.

The Application Functions frequently create, read, update or delete Data Objects. Thus does Application Function confidentiality and integrity affect the same properties of the Data Objects, and it also brings that the accuracy of the Data Objects causally affect the accuracy of Application Services.

Further down in the structure, are Application Functions supplied with Data Objects through Infrastructure Services, for instance by means of a Database Management System (DBMS). The availability, confidentiality and integrity of the Infrastructure Services impact the same properties of the Application Functions. To be able to represent which Infrastructure Service affects which Data Object (through the Application Functions) an “Assign” relation has been introduced into the abstract model.

The functionality of Infrastructure Services is physically implemented in Nodes. Nodes can be either Devices, i.e. the hardware, or System Software which is the needed infrastructure software. Nodes can consist of sub-nodes, as for instance when a Device in the form of a server runs System Software in the form of an operating system.

The availability, confidentiality and integrity of the Node will affect the same properties of the Infrastructure Service and the overall Node availability, confidentiality and integrity is affected by System Software and Device availability, confidentiality and integrity. Nodes, Devices and System Software all feature inherent availability, confidentiality and integrity.

Finally, Nodes exchange information over Communication paths which are physically realized by Networks. Whenever the Networks are unavailable, or become compromised with respect to confidentiality or integrity the Infrastructure Services seize to function or become compromised. Communication paths are abstract concepts without inherent properties. Rather the availability, confidentiality and integrity of the Communication Path are identical to those of the Networks of which it consists. Finally, Networks, have an inherent availability, confidentiality or integrity.

6. The case study

A case study with the purpose of evaluating two potential Service Oriented Architecture (SOA) scenarios has been performed. The evaluation was using the abstract model for the system quality analyses described in the previous section. The scenarios originate from a joint project between a leading European power company and the Technical University of Berlin. The project ran between autumn 2007 and spring 2008. The purpose of the project was to gather experiences of SOA solutions to provide input to a decision regarding the company’s future architectural strategy regarding SOA. Specifically, the project wanted to evaluate the quality of different SOA infrastructures.

To achieve this, the same process has been implemented twice using different SOA infrastructures. The project chose the sales business process since it has suitable complexity. It differentiates between more than 40 activities supported by a range of diverse stakeholders.

Due to the project’s focus on technical issues, no effort was spent on re-engineering the business process. The already existing technical infrastructure and business applications were encapsulated into services. The applications were a mixture of customized SAP modules and proprietary applications. Altogether 50 services were implemented.

In the end, only one Application Service called “Check Customer Address” was investigated. This choice was made to be result efficient since it was assumed that the results would generalize to the rest of the services in the scenarios.

6.1. The scenarios

The different SOA infrastructures are using extended service repositories. The first scenario uses the commercial software Sun JCaps [22] see Figure 5. The second scenario is based around research driven solution called PrOSeRO - Process Oriented Service Repository & Ontology, see Figure 4. The PrOSeRO project is funded by Deutsche Telekom Laboratories and has been partly published in [23] and [24]. Both scenarios mainly used open source software where available.

The scenarios were identical with respect to hardware Devices and Networks. Both scenarios used virtual servers, represented as System
Figure 4: The concrete model for the PrOSeRO scenario with collected input data.

Figure 5: The concrete model for the Sun JCaps scenario with collected input data.
Software in the concrete model, to operate components such as databases, application servers and the infrastructure for the JavaServerPages (JSP) user interface [25].

Due to different features provided by the two approaches, “Check Customer Adress” is orchestrated using three further Application Functions in the JCaps scenario and four Application Functions in the PrOSeRO scenario.

To enable the underlying services both scenarios provide different Infrastructure Services based on different System Software as shown in the concrete models in Figure 4 and Figure 5. In the JCaps scenario an Oracle Database was used, and the PrOSeRO scenario relied on a MySQL database. Further, the JCaps scenario implemented its application server on Sun’s Logical Host whereas PrOSeRO used a Tomcat server. Finally, the orchestration engines were implemented using Sun eVision and the Active BPEL respectively.

6.2. The data collection

The collected system quality data for the classes Device, System Software (Virtual Server) and the Network are identical for both scenarios. These figures were estimated by the IT operations department’s personnel with aid of their infrastructure management system. The availability of statistical data and the IT operations department’s longlasting experiences ensures a high accuracy of the data that was collected concerning the infrastructure entities

The data on the scenario-specific System Software of the JCaps scenario and the Application Component class in both scenarios are taken from relevant Service Level Agreements. Service Level Agreements only contain the lower bounds for the respective system qualities and the numbers concerning these entities may therefore be higher than specified in the SLA’s.

For both scenarios the availability, accuracy, confidentiality and integrity figures concerning the Application Functions, the System Software Active BPEL and Sun eVision, and the Data Objects have been estimated by project members based on experience and previous test results. Accuracy data concerning Data Objects and Application Functions were estimated by project members. All of these estimations suffer from a rather high uncertainty since the project members were asked to estimate data after project completion and did not consider these aspects during project execution.

6.3. The results

The concrete models were inserted as Bayesian networks and into the GeNIe tool, which can be used to graphically build and compute Bayesian networks. Excerpt screenshots from GeNIe for the availability analyses of the respective scenarios can be seen in Figure 7 below. The overall results of the analyses are shown in Figure 6 below.

According to the computations, the Sun JCaps scenario availability was 81 % which significantly surpassed the availability of the PrOSeRO scenario which only attained 65 % availability. The main reason for this difference stemmed from the orchestration engine, where the PrOSeRO Active BPEL engine featured a much lower availability than its counterpart orchestration engine in the JCaps scenario. This is consistent with the experiences of running the systems, where SUN JCaps was considered more stable.

With respect to accuracy, the JCaps scenario was slightly better due to the fact that fewer Application Functions were needed to orchestrate the Application Service and that the Oracle Database features validation checks on Data Objects thereby increasing their accuracy.

The JCaps scenario was also slightly better than the PrOSeRO scenario with respect to confidentiality and integrity, mainly due to the higher security of the Oracle Database compared to the MySQL-database.

![Figure 6: The results of the evaluation.](image)

7. Conclusions

EA frameworks and approaches are today focused heavily on how to model, but oftentimes lacks support for performing analyses on the models. This paper is showing how the ArchiMate EA metamodel can be extended to support performing enterprise system
quality analyses. By complementing the ArchiMate metamodel with attributes and causal relations, modeled in Bayesian networks, between those attributes, the metamodel is converted into a so called abstract model.

Although ArchiMate was the framework of choice for this paper, the method of extending architecture metamodels with attributes and attribute relations is general and can be applied to all architecture metamodels.

In an example we show how an abstract model for system service quality analysis is used for evaluating two competing service oriented architecture scenarios with respect to four system qualities; accuracy, availability, confidentiality and integrity.

The types of analyses described in this paper are not confined to the infrastructure or application domain. Would one want to include for instance the IT organizations impact on the application’s availability this is fully possible. Furthermore, the choice of analyses performed using our approach, i.e. system qualities in this case, is not limited to address merely system related issues. One has the choice of use the method for analysis of for instance business IT alignment, or IT organization maturity.

Figure 7: GeNle screenshot for the availability assessment of the PrOSeRO scenario
8. References (1p)