Softw Syst Model (2014) 13:1085–1116
DOI 10.1007/s10270-012-0288-2

THEME SECTION PAPER

An enterprise architecture framework for multi-attribute information systems analysis

Per Närman · Markus Buschle · Mathias Ekstedt

Received: 11 September 2011 / Revised: 7 August 2012 / Accepted: 13 September 2012 / Published online: 16 November 2012

Abstract Enterprise architecture is a model-based IT and business management discipline. Enterprise architecture analysis concerns using enterprise architecture models for analysis of selected properties to provide decision support. This paper presents a framework based on the ArchiMate metamodel for the assessment of four properties, viz., application usage, system availability, service response time and data accuracy. The framework integrates four existing metamodels into one and implements these in a tool for enterprise architecture analysis. The paper presents the overall metamodel and four viewpoints, one for each property. The underlying theory and formalization of the four viewpoints is presented. In addition to the tool implementation, a running example as well as guidelines for usage makes the viewpoints easily applicable.

Keywords Enterprise architecture · Enterprise architecture analysis · Enterprise architecture tool · Data accuracy · Technology usage · Service availability · Service response time

1 Introduction

The use of IT is pervasive in today's business and significantly impacts business operations [11,26,79,83]. Managing IT properly has become an imperative and enterprise architecture (EA) has become an established discipline to this end [10,15,104,105,119]. EA uses architecture models to aid communication and ease understanding of complex systems comprising multiple business, application and IT infrastructure components [60]. Analysis of architecture models for decision support is one tenet of such understanding [60,68]. EA analysis involves querying models with the aim of evaluating various properties, such as business IT-alignment, security and more. Nevertheless, the concept of EA analysis has not made it into mainstream enterprise architecture practice. There are some publications on the topic of EA analysis, notably [24,34,40,51,60,61], but the major EA frameworks such as the Zachman framework [131], TOGAF [115], NAF [89] or DoDAF [30] fail to address the topic at all.

EA analysis is useful both in identifying improvement areas related to the as-is architecture but also when faced with decisions regarding the future to-be architectures [61]. The process of using EA analysis for decision-making involves (i) defining the scenarios—for instance the choice of a best-of-breed application or a component from an ERP-package; (ii) determining the properties of interest when making the decision—for instance functionality, security or availability; (iii) modeling the scenarios using an architecture metamodel which allows for analysis of the properties of interest; (iv) analyzing the scenario's properties of interest to the decision-maker—functionality and availability-wise, the best-of-breed application may be superior to the ERP package but does not meet the mandated security levels; and (v) make a decision—for instance the ERP package may be chosen since security is prioritized more highly than the other properties.

This paper presents the results from a research program on quantitative EA analysis for decision-making. The approach employs probabilistic modeling and analysis [60,69,111,116]. A specific aim has been to create models which are

Communicated by Dr. Tony Clark, Balbir Barn, Alan Brown, and Florian Matthes.

P. Närman (✉) · M. Buschle · M. Ekstedt
Royal Institute of Technology, Industrial Information and Control Systems, Osquldas väg 10, 10044 Stockholm, Sweden
e-mail: pern@ics.kth.se

not prohibitively expensive to use, especially in terms of data collection; a frequent cost driver [58,88].

As a part of this research program, four metamodels based on ArchiMate [114] have been developed for the analysis of service availability [90], response time [1][85], data accuracy [87] and application usage [86].

Application usage, i.e., determining whether application users voluntarily use the application, is a key concern in evaluating application portfolios since low application usage is often associated with poor user performance [25,26,128]. Service response time, i.e., the time a service requires to complete a transaction, significantly impairs user experience when degraded [93]. Service availability, i.e., the fraction of the total time that a service is available to its users, is crucial to ensure continuous business operations [107]; not only are the direct costs of unavailable IT systems high [52], but IT incidents disrupting business operations also have an adverse impact on the market value of publicly traded companies [8]. Poor data accuracy, i.e., the fraction of a set of data objects are accurate, impairs organizational decision-making, drives cost of operations and reduces customer satisfaction [99].

The four properties do not constitute an exhaustive set of aspects to consider when making IT-related decisions. The reason for choosing these particular properties is due to requirements made by organizations with which the authors have performed case studies over the past years (notably the one reported in [38]). Even though the properties are not the full set of important properties, they still cover many important facets of information systems. This can be shown by comparing the attributes to Delone and McLean's model for information system success [25,26]. This model posits that system quality, service quality and information quality affect user satisfaction, user intention to use as well as system usage and the higher these factors are the higher the net benefits will be [26]. The application usage metamodel could be mapped to the user satisfaction, intention to use and system usage. The service availability and the service response time metamodels are squarely within the service quality dimension and data accuracy is relevant to the information quality dimension.

It is beneficial to reuse models as far as possible since it promotes easier communication between different domains [77] and keeps modeling costs down [95]. Creating multiple and incompatible models depicting the same reality but with a slightly different purpose is a common problem. For instance, process models made by for the purpose of ISO 9000 certification are seldom reused in "regular" EA modeling for purpose of design or documentation [95]. Despite using ArchiMate as their base metamodel, the four metamodels were not fully consistent in their use of constructs thus prohibiting architects from leveraging the architecture content for multiple analyses and driving the cost of modeling.

Another important aspect of information system decision-making is the necessity to make trade-offs between different properties. For instance, security is much improved by adding anti-virus software, but it may have an adverse impact on performance.

This paper integrates these four metamodels on EA analysis into one combined EA metamodel thereby allowing both re-use and trade-off analysis. The four different metamodels mentioned above are codified as pre-defined and re-usable viewpoints each addressing a specific concern, i.e., a specific analysis.

Moreover, the original metamodels were expressed using a probabilistic formalism known as probabilistic relational models (PRM) featuring Bayesian networks [37]. PRMs are capable of integrated modeling and analysis but come with a number of drawbacks including (i) intractability of inference—when dealing with particularly hybrid Bayesian networks, the analyst may encounter difficulties in performing accurate inference [20], this almost always necessitates approximate reasoning; (ii) no mechanism to query models for structural information—the PRMs are limited to reasoning about object attributes and finally; (iii) poor support for specifying modeling constraints—PRMs are not very sophisticated in defining modeling constraints.

To overcome these drawbacks, the four metamodels have been re-implemented in the more expressive formalism known as the p-OCL, short for probabilistic Object Constraint Language [116], which extends OCL [1] with probabilistic reasoning. The viewpoints have also been implemented in a tool (known as EAAT for Enterprise Architecture Analysis Tool [2]) for p-OCL modeling and analysis [16]. The resulting EAAT files can be found online and downloaded, see Appendix C.

In summary, this article has the following aims:

1. To integrate four previously published metamodels for architecture analysis into one coherent metamodel thus allowing reuse of models and easy trade-off analysis.
2. To formalize the four metamodels using the p-OCL formalism and to implement them in the EAAT tool.
3. To present the individual metamodels as viewpoints pertaining to the overall metamodel.

The result, an integrated metamodel, is a Design Theory [42] or a theory for "design and action". Unlike other scientific theories which aim to predict or analyze certain aspects of reality, a theory for design and action provides guidance

[1] Pre-print, revision submitted to the Journal of Strategic Information Systems, manuscript available for review upon request.

[2] http://www.ics.kth.se/eat.

on how to do something (in this case perform EA analysis) [42]. It will be discussed as such in the Discussion section.

The remainder of this article is outlined as follows. Section 2 will cover some related works on EA analysis. Section 3 will briefly introduce the OCL formalism in which the metamodel is expressed. Section 4 provides an overview of the entire metamodel. Section 5 presents the four viewpoints in detail, Sect. 6 discusses the results from a design science perspective and Sect. 7 concludes the paper.

2 Related works

The EA frameworks in use today offer little or no support for architecture analysis. The Zachman framework categorizes architecture models according to a taxonomy, but does not attach them to any metamodel and does not offer any support as far as analysis goes. The military frameworks MoDAF [81] and DoDAF [30] both offer wide arrays of viewpoints addressing multiple stakeholder concerns. They do not, however, formally specify how to perform analysis beyond offering modeling suggestions. An exception to the rule is the LISI framework for interoperability analysis which is tightly integrated with the DoDAF framework [63].

The ArchiMate EA framework [114] comes with extensions which allow for EA analysis, specifically of performance and IT cost [50,51]. The former is partly integrated in the framework presented here. However, there is a lack of formal integration between the metamodel and the analysis mechanisms which makes tool implementation difficult.

There is a stream of research on EA analysis of individual non-functional properties [61] such as security [111], modifiability [69], interoperability [117] or data accuracy [87]. These metamodels presented in these examples have however been limited to analysis of single properties. An early attempt at EA analysis of multiple properties was presented in the work by Gammelgård et al. [38], but with a very weak link to actual architecture models.

The work by [40] proposed describing the EA of an entire enterprise and perform simulations on it to identify opportunities to increase enterprise profitability. Although worthy of praise for its ambition, the notion of simulating the entire enterprise has very little connection to IT decision-making. De Boear et al. [24] present an XML-based formalism for EA analysis, but offer very few details on how specifically to undertake the analysis. Enterprise architecture patterns is a topic gaining traction in the community, but these either support qualitative analysis [32] or focuses only on business processes [106].

As far as tools are concerned, the ABACUS tool [31] offers several analysis possibilities including performance, total cost of ownership and reliability analyses. Although there is some overlap with the present work in terms of addressing concerns, (performance and availability) the ABACUS does not offer any support for application usage or data accuracy analysis.

There are plenty of multi-attribute software architecture analysis methods such as [5,7,39,64–66]. These are, however, demanding in terms of data collection and often require the user to exhaustively test the system's constituent components which is not feasible in an EA context with a large number of components.

There are many frameworks for assessment of application portfolio evaluation [13,102,108,128]. None of these, however, are able to explain why some applications are voluntarily used and why others are not.

The software architecture community offers specific reliability and availability analysis frameworks as well such as [9,21,56,73–75,101,103,126,127,130], but these too are rather cumbersome to use in an EA setting. There are some dedicated EA availability analysis methods, but these are purely qualitative [97], fail to take component redundancy into account [14,47], or restrain themselves to the military domain [35,36].

As for response time, there are three kinds of analysis methods: [57] *measurements*—using experimental methods to directly measure response time [82], *simulation-based methods*—creating executable response time simulation models [5,31,80], and *analytical modeling*—using queueing theory [12,17,18,70,100] to measure response time [96,110]. There are methods for response time analysis of business processes (e.g., [2]), software applications (e.g., [110]), the infrastructure domain (e.g., [43]), or embedded systems (e.g., [27]), however, there are few attempts at integrating these perspectives into one coherent method [49]. Both the IT governance framework COBIT [55] and service management framework ITIL [118] propose capacity management processes service response time management but do not go into detail on how to perform response time analysis. The method by Iacob and Jonkers [49–51,71] employs queuing networks to analyze performance incorporating all architectural domains.

As for data accuracy, The Quality Entity Relationship (QER) model and the Polygen model represent some of the earliest attempts at classifying data quality [124] using relational algebra but without integrating the analysis with modeling. So called Information Product Maps models [76], extended to an UML profile [125], are able to graphically depict information flow but without quantitative data quality analysis. Ballou et al. [4] used data flow diagrams and a quantitative approach to illustrate data accuracy deterioration in applications, but this was confined to numerical data. Cushing et al. [22] presented a method to illustrate how all kinds of data could both improve and deteriorate across business processes, but not in a modeling context.

In conclusion, we find that there are no methods available to perform integrated EA analysis of the four properties mentioned in the introduction, and we therefore proceed to integrate the metamodels published in [85–87,90].

3 Probabilistic OCL

The Object Constraint Language (OCL) is a formal language typically used to describe constraints on UML models [1]. These expressions typically specify invariant conditions that must hold for the system being modeled, pre- and post-conditions on operations and methods, or queries over objects described in a model.

The p-OCL language is an extension of OCL for probabilistic assessment and prediction of system qualities, first introduced in [116] (under the name Pi-OCL). The main feature of p-OCL is its ability to express uncertainties of objects, relations and attributes in the UML-models and perform probabilistic assessments incorporating these uncertainties, as illustrated in [116].

A typical usage of p-OCL would thus be to create a model for predicting, e.g., the availability of a certain type of application. Assume the simple case where the availability of the application is solely dependent on the availability of the redundant servers executing the application; a p-OCL expression might look like this,

```
context Application:
  attribute available : Boolean =
  self.server->exists(s:Server|s.available)
```

This example demonstrates the similarity between p-OCL and OCL, since the expression is not only a valid p-OCL expression, but also a valid OCL expression. The first line defines the context of the expression, namely the application. In the second line, the attribute `available` is defined as a function of the availability of the servers that execute it. In the example, it is sufficient that there exists one available server for the application to be available.

In p-OCL, two kinds of uncertainty are introduced. Firstly, attributes may be stochastic. When attributes are instantiated, their values are thus expressed as probability distributions. For instance, the probability distribution of the instance *myServer.available* might be

```
P(myServer.available)=0.99
```

The probability that a `myServer` instance is available is thus 99 %. For a normally distributed attribute `operatingCost` of the type Real with a mean value of $3,500 and a standard deviation of $200, the declaration would look like this,

```
P(myServer.operatingCost)=Normal(3500,200)
```

i.e., the operating costs of server is normally distributed with mean 3,500 and standard deviation 200.

Secondly, the existence of objects and relationships may be uncertain. It may, for instance, be the case that we no longer know whether a specific server is still in service or whether it has been retired. This is a case of object existence uncertainty.

Such uncertainty is specified using an existence attribute `E` that is mandatory for all classes (here using the concept class in the regular object-oriented aspect of the word), where the probability distribution of the instance `myServer.E` might be

```
P(myServer.E)=0.8
```

i.e., there is a 80 % chance that the server still exists.

We may also be uncertain of whether `myServer` is still in the cluster servicing a specific application, i.e., whether there is a connection between the server and the application. Similarly, this relationship uncertainty is specified with an existence attribute `E` on the relationships.

In the present article the reader will be confronted with p-OCL in three ways: (i) in the form of metamodel *attribute specifications*; (ii) as metamodel *invariants* which constrain the way in which the model may be constructed; and (iii) as *operations* which are methods that aid the specification of invariants and attributes. Appendix A and B contains all of these expressions.

An example metamodel attribute expression is shown below:

```
context UsageRelation
attribute self.ApplicationWeight: Real =
getFunctionality()/isAffected_1_inv.
    use_5_inv.Functionality
```

This is referring to the class `UsageRelation` in Fig. 2 and specifies that `getFunctionality()` operation should be utilized. The operation getFunctionality() is specified as follows:

```
context UsageRelation
operation getFunctionality(): Real =
self.isAffected_2.assigned_inv->select(
    oclIsKindOf(ApplicationFunction)).
    oclAsType(ApplicationFunction).
    Functionality->sum()
```

where it says that `getFunctionality()` requires no input, and generates a `Real` as output according to a statement (which will be explained in Sect. 5.1). See also Statement 2 in Appendix A.

An example invariant, `noWriteAndRead`, can be found below:

```
context InternalBehaviorElement
invariant noWriteAndRead =
not(read_Function_inv->exists(do:
    PassiveComponentSet| write_Function->
    includes(do)))
```

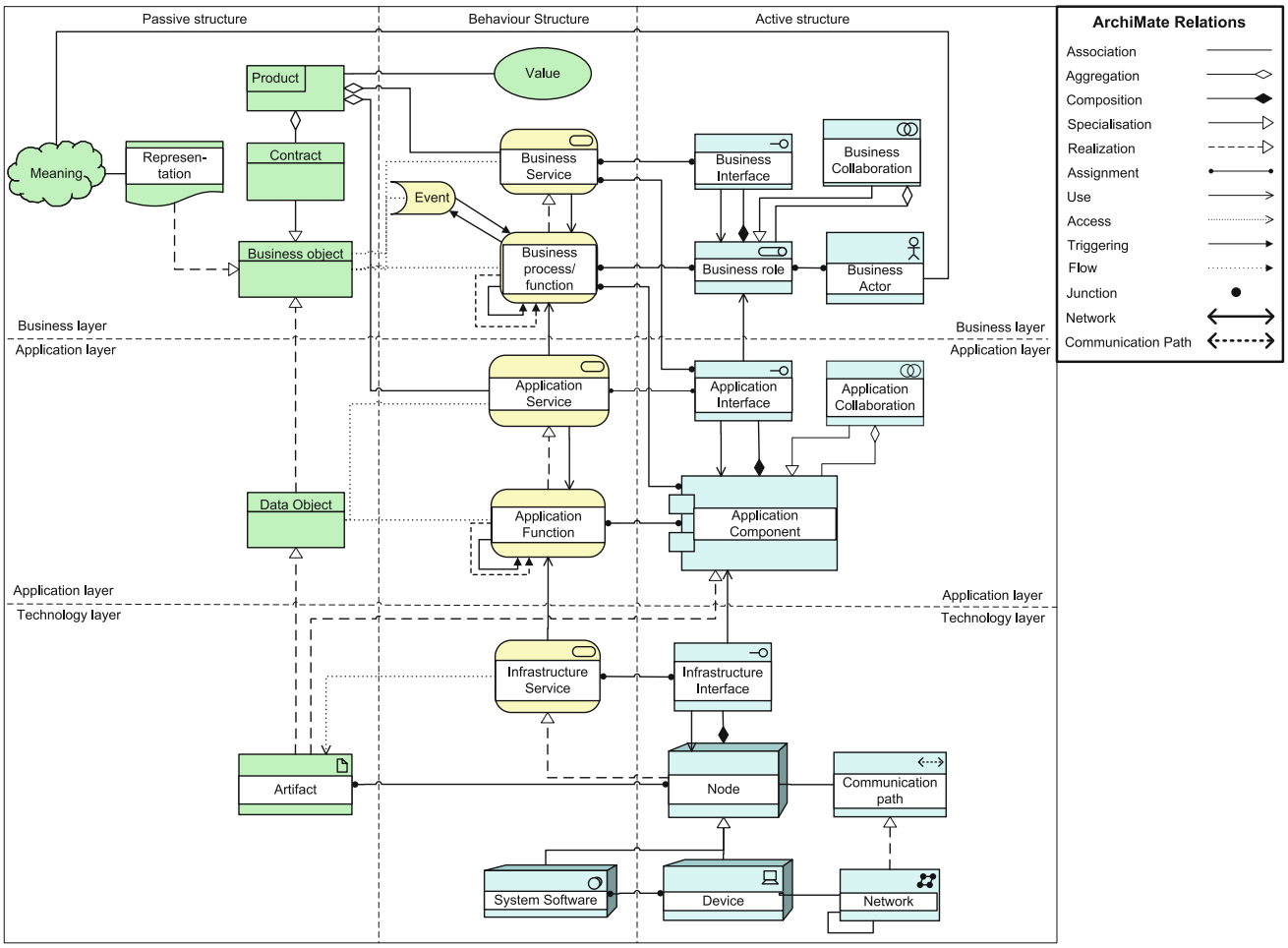


Fig. 1 The original ArchiMate metamodel [114]

This specifies that objects of the class `InternalBehaviorElement` from Fig. 2 are not allowed to both write and read the same data object.

A full exposition of the p-OCL language is beyond the scope of this paper. Suffice to say here that the EAAT tool described in [16] now implements p-OCL using the EMF-OCL plug-in to the Eclipse Modeling Framework [33] and has been employed to implement the metamodels of this paper. The probabilistic aspects are implemented in a Monte Carlo fashion: in every iteration, the stochastic p-OCL variables are instantiated with instance values according to their respective distribution. This includes the existence of classes and relationships, which are sometimes instantiated, sometimes not, depending on the distribution. Then, each p-OCL statement is transformed into a proper OCL statement and evaluated using the EMF-OCL interpreter. The final value returned by the model when queried is a weighted mean of all the iterations.

4 Metamodel

This section will describe the overall metamodel which underlies the viewpoints described in the next section.

4.1 ArchiMate

The metamodel is a modification of the ArchiMate metamodel [114] which is a mature and much used EA framework, see Fig. 1.

The original ArchiMate metamodel contains *active structure* elements, *passive structure* elements and *behavioral structure* elements. Behavioral elements describe dynamic behavior which can be performed by either IT systems or human beings and which are modeled as active structure elements. ArchiMate differentiates between internal behavior elements, which are directly linked to active structure elements, and external behavior elements i.e., different kinds of services, which represent the behavior as seen

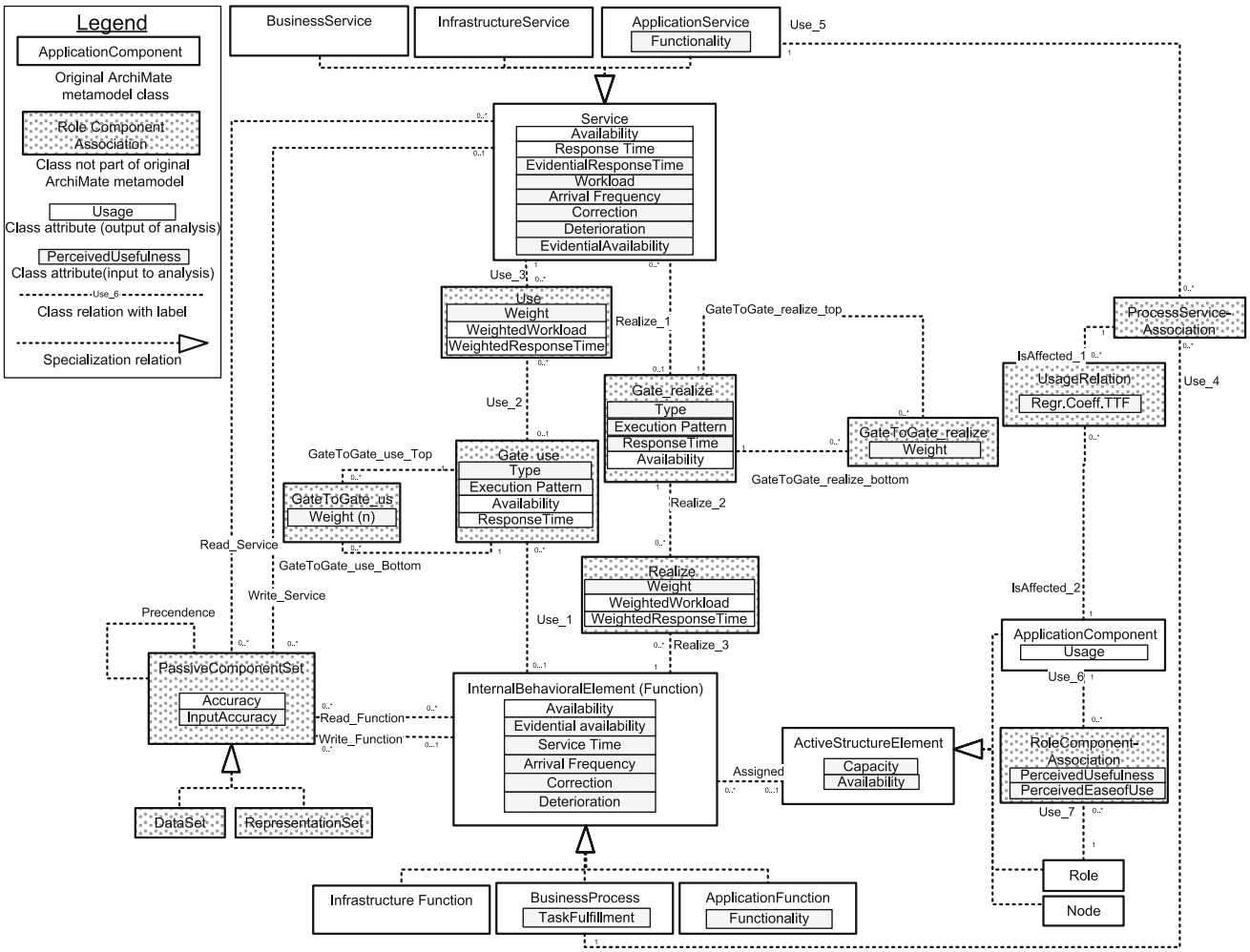


Fig. 2 The metamodel on which the viewpoints are based. The *white* attributes show the output attributes, and the *grey* denote the input attributes; those that need to be set by the users. The *grey boxes* represent classes which have been added to the original ArchiMate metamodel

by the users. The passive structure elements describe what is accomplished as a consequence of the behavior. Examples of behavior concepts are `BusinessProcesses` or `ApplicationServices`, active structure components can be `ApplicationComponent` or `Roles` and examples of passive structure elements are `DataObjects`.

Furthermore, the classes of the ArchiMate metamodel can be grouped in three layers; business, application and technology. This is fairly standard in EA modeling apart from the fact that ArchiMate integrates the informational aspect in all three layers, information is otherwise often considered as a layer of its own, see e.g., TOGAF 9 [115].

4.2 The metamodel

The ArchiMate metamodel has been augmented with a number of classes and attributes, and this has resulted in the metamodel described in Fig. 2. The metamodel shows all of the metamodel classes, and all attributes which are relevant to users by either requiring some sort of input from the user (grey attributes) or by providing the users with useful information about either one of the four properties (white attributes). For clarity, the metamodel in Fig. 2 omits attributes which are used as intermediary variables, these are included in the individual viewpoint below. Notice also that the present metamodel contains the `InfrastructureFunction` which was not present in the original metamodel, but added in more recent ArchiMate works [72].

The passive structure elements of the metamodel are called `DataSet` and `RepresentationSet` and are slight alterations of the original ArchiMate `DataObjects` and `Representations`. The modification consists in defining `DataSet` and `Representation Set` as sets comprising multiple `DataObjects` or `Representations`. These are used for the data accuracy viewpoint.

The internal active structure elements consist of `BusinessProcesses`, `ApplicationFunctions`

and InfrastructureFunctions. These interface with the external services through a number of placeholder classes, Realize and Use which allow the modeler to set attribute values on relations for the service response time viewpoint, and Gate_Use and Gate_Realize which are logical gates depicting how availability flows through the architecture in the availability viewpoint. The GateToGate_Realize and GateToGate_Use classes are used as containers for intermediate attributes when multiple gates are connected which is important in the service availability and the response time viewpoints.

The external services of the metamodel are ApplicationService and Infrastructure Service. ApplicationServices and Business Processes have a special relation as shown in the Process-ServiceInterface class which is used in the Application Usage viewpoint.

The active structure elements of the metamodel are represented as Roles, Application Components and Nodes. Roles interface with Application Components through the RoleComponentInterface class for the Application Usage viewpoint. Furthermore, there is a class between the Process-Service Interface class and the ApplicationComponent class which is called UsageRelation. This is also used in the Application Usage viewpoint.

4.3 Creating the metamodel

As mentioned above, the metamodel was created by integrating and slightly altering the original metamodels from [85–87,90]. The metamodel classes were compared one-by-one and those classes which were identical between the metamodels had their respective sets of attributes and relations merged. In some cases, changes to the overall metamodel had to be made. In particular, the service response time metamodel and the availability metamodel exhibited traits which were at odds with each other: firstly, the response time metamodel differentiated Services from InternalBehaviorElements. The availability metamodel, however did not distinguish between Service and InternalBehaviorElement. Secondly, the two metamodels used logical gates in a slightly different manner. Ultimately, these two differences led to the inclusion of both the Service and InternalBehaviorElement classes as well as two new logical gates, one for the Use relation and one for the Realize relation.

The original response time metamodel stipulated that there was a one-to-one relation between Internal BehaviorElements and Services. This is incompatible with the application usage and service availability metamodels, and thus some changes had to be made to the service response time metamodel to accommodate many-to-many relations (see Sect. 5.3).

Furthermore, the original metamodels were expressed using either the Probabilistic Relational Model (PRM) [37] or the hybrid PRM (HPRM) [84] formalism, implemented in the EAAT tool. The present metamodel is also implemented in the EAAT tool, but uses the p-OCL formalism instead. Thus, all of the PRM and HPRM expressions have been transformed into p-OCL statements. In the case of the logical gates, the superior flexibility of the p-OCL formalism made it possible to avoid having separate metamodel classes for AND-gates and OR-gates, and instead choose the kind of gate in the attribute Gate.Type.

An addition to the previous work are the p-OCL invariants that express constraints on how model objects may be connected to each other; these are found in Appendix B. There is the checkLayerMatching invariant (Statement 23), which specifies that Services may only be realized by InternalBehaviorElements from the same ArchiMate layer, there is the noReadAndWrite and noWriteAndRead invariances (Statements 21 and 22), which make sure that a PassiveComponentSet cannot be read and written simultaneously by the same Service or InternalBehaviorElement, respectively.

5 Viewpoints

This section will present a number of *viewpoints*, i.e., re-usable ways in which to model so as to address commonly encountered *stakeholder concerns* [113]. Each viewpoint describes the specific concerns it addresses and the stakeholders likely to possess these concerns. Next is an account of the theory used to perform the analysis that addresses the concerns together with a detailed description of how the viewpoint works, an example *view* is presented as well as some guidelines for how to use the viewpoints in practice. TOGAF 9 describes the relation between viewpoints and views: "A viewpoint is a model (or description) of the information contained in a view" [115]. Conversely, the view is a concrete representation of reality described according to a viewpoint. Throughout the text, there will be qualitative definitions of all derived viewpoint attributes together with references to the actual OCL Statements found in Appendix A.

5.1 Application usage viewpoint

The application usage viewpoint is an adaption of the metamodel which was developed and validated in [86].

Concerns The first viewpoint concerns application usage; why do users voluntarily embrace certain applications and object to using others? Voluntary application usage is a very

important indicator of the quality of the application portfolio [26,128].

Stakeholder The stakeholders are those interested taking a top-down perspective on the application portfolio. These may include enterprise architects or application architects and ultimately the organization's CIO.

Theory Two of the most widely used theories for technology and IT usage predictions are the technology acceptance model (TAM) [23] and the task-technology Fit (TTF) model [41].

The TAM posits that the usage of information systems can be explained by two variables; the perceived usefulness (PU) and the perceived ease of use (PEoU) of the information system [23].

TTF is built on the idea that if the users perceive a information systems to have characteristics that fit their work tasks, they are more likely to use the technology and perform their work tasks better. Dishaw and Strong [28] defined task-technology fit as *"the matching of the functional capability of available software with the activity demands of the task"*, and operationalized task and tool characteristics for the domain of computer maintenance based on previously published reference models of computer maintenance tasks [122] and maintenance software tool functionality [44]. Dishaw and Strong [28] used the concept of strategic fit as interaction [121] (meaning multiplication of task and functional fulfillment) to operationalize TTF.

Dishaw and Strong [29] also applied an integrated TAM/TTF model with greater explanatory power than the separate models, something which was corroborated by [19, 67,92]. Similarly, the application usage viewpoint employs a combined TAM/TTF theory.

Viewpoint description This viewpoint needs to be tailored to its application domain. The tailoring involves defining and operationalizing the domain's tasks, IT functionality, how the IT functions support the tasks (i.e., the TTF variables) and finally the quantitative degree to which this support exists in the form of a linear regression model. Närman et al. [86] presented a tailored metamodel and associated linear regression model for the maintenance management domain.

The viewpoint is presented in Fig. 3. The aim of employing the viewpoint is to obtain a value for the `ApplicationComponent.Usage` attribute. This is

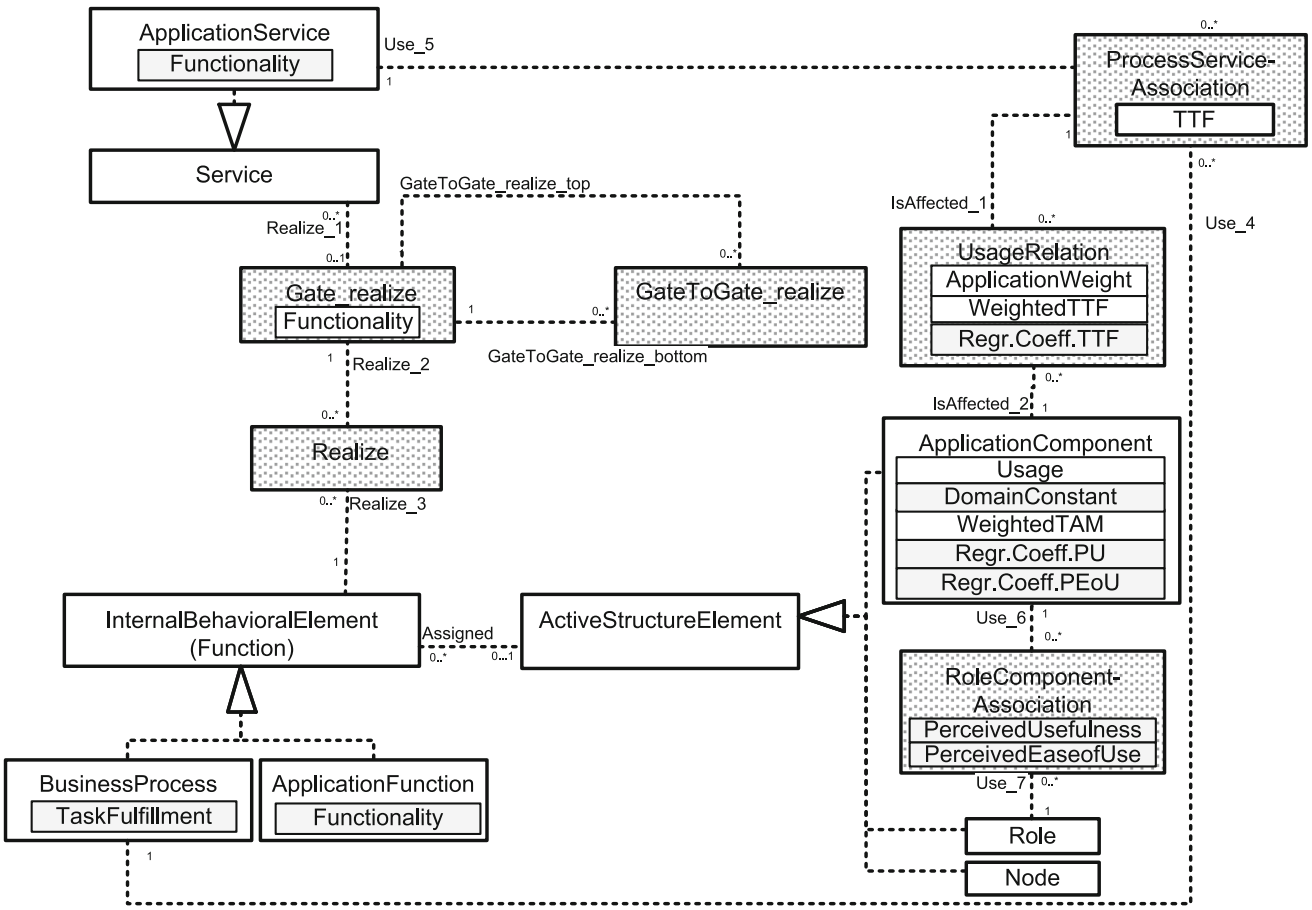


Fig. 3 The application usage viewpoint

derived through a linear regression model which relates the pertinent TTF and TAM variables to usage according to Eq. 1.

$$\text{Usage} = (\alpha + \beta_1 * TTF_1 + \cdots + \beta_n * TTF_N + \beta_{n+1} * PU + \beta_{n+2} * PEoU) \quad (1)$$

The constants α and $\beta_{i,\ldots,n+2}$ (one β per TTF matching, see below) are constants determined by processing empirical survey data of application usage, PU, PEoU and TTF for the domain in question.

To express attributes on relations, a number of placeholder classes are introduced. Thus, the viewpoint features `RoleComponentInterface`, which can be used to express the PU and PEoU attributes on the relations between `Roles` and `ApplicationComponents`. The `ProcessService Interface` is used to determine the degree to which TTF exists between pairs of `BusinessProcesses` and `ApplicationServices`. The `UsageRelation` placeholder class is used to propagate TTF values back to the `ApplicationComponents` which implements (parts of) the functionality.

The PU and PEoU variables are found as the attributes `RoleComponentInterface. PerceivedUsefulness` and `RoleComponentInterface.PerceivedEaseOfUse` (PEoU and PU henceforth), respectively. These are assessed by asking the actors using `ApplicationComponents` about the usefulness and ease of use of that particular application. The attribute values are derived by taking the mean of the answers per `Role` and `ApplicationComponent` pair.

The attribute `ApplicationComponent.WeightedTAM` is the linear combination of the mean of all `RoleComponentInterface.PU` and `RoleComponentInterface.PEoU` attributes pertaining to the `ApplicationComponent`, and the regression coefficients, in the model denoted `ApplicationComponent.Regr.Coeff.PU` and `ApplicationComponent.Regr.Coeff.PEoU`, respectively, see Statement 1.

As mentioned already, the TTF values need to be assigned to `ApplicationComponents` through the `UsageRelation`. There are three attributes on the `UsageRelation` class: (i) `UsageRelation.RegressionCoefficient`, (ii) `UsageRelation.ApplicationWeight` and (iii) `UsageRelation.WeightedTTF`. These express (i) the regression coefficient showing the quantitative impact a particular TTF variable has on `Application Component.Usage` (the β in Eq. 1), (ii) the relative functional contribution of the `ApplicationComponent` to the total `ApplicationService.Functionality`, and (iii) the TTF value weighted using the former two attributes (see Statement 3).

The TTF variable itself is represented as the `ProcessServiceInterface.TTF` attribute and is derived by multiplying the attributes `ApplicationService.Functionality` and `BusinessProcess.TaskFulfillment`, see Statement 5. `BusinessProcess.TaskFulfillment` is the mean of user assessments of the task fulfillment for a particular business process, and `ApplicationService.Functionality` is the mean of user assessments of the total offered functionality with respect to a standard service description.

This functionality may be offered by several `ApplicationComponents`; the specific functionality implemented in a particular `ApplicationComponent` is therefore modeled in the `ApplicationFunction.Functionality` attribute. Sometimes several `Application Functions` aggregate to form other `Application Functions`. The sum of their functionality is then found in the `ApplicationService.Functionality` attribute. The `ApplicationFunction.Functionality` attribute is also used to determine the `UsageRelation.ApplicationWeight` by dividing the associated `ApplicationFunction.Functionality` and `ApplicationService.Functionality` attributes, see Statement 2.

Finally, the sum of the attributes `ApplicationComponent.WeightedTAM`, the `UsageRelation.WeightedTTF` and `ApplicationComponent.DomainConstant` (the α in Eq. 1) yields the `ApplicationComponent.Usage` attribute, see Statement 4.

Validation of the technology usage viewpoint The technology usage viewpoint was validated by (i) demonstrating that it was possible to tailor the viewpoint for a specific domain, and (ii) by showing that the operationalization of the tailored viewpoint did indeed account for variations in application usage within the maintenance management domain. As for (i) it was taken care of by creating reference models of IT functionality and task descriptions based on [53,59,62,94,109,129] and validating these with interviews.

The second part of the validation consisted of testing the three task-technology fit dimensions in a survey with 55 respondents working with maintenance management at five companies. The results showed that the model taken as a whole did predict a high degree of variation in application usage (adj. $R^{2**} = 0.548$).

More details on the maintenance management operationalization and the validation of the viewpoint can be found in [86].

Guidelines for use In the case the organization does not have reference models for tasks and functionality, these have to be developed, perhaps using the approach of [86]. Once the appropriate models are in place, however, the viewpoint may be employed as follows:

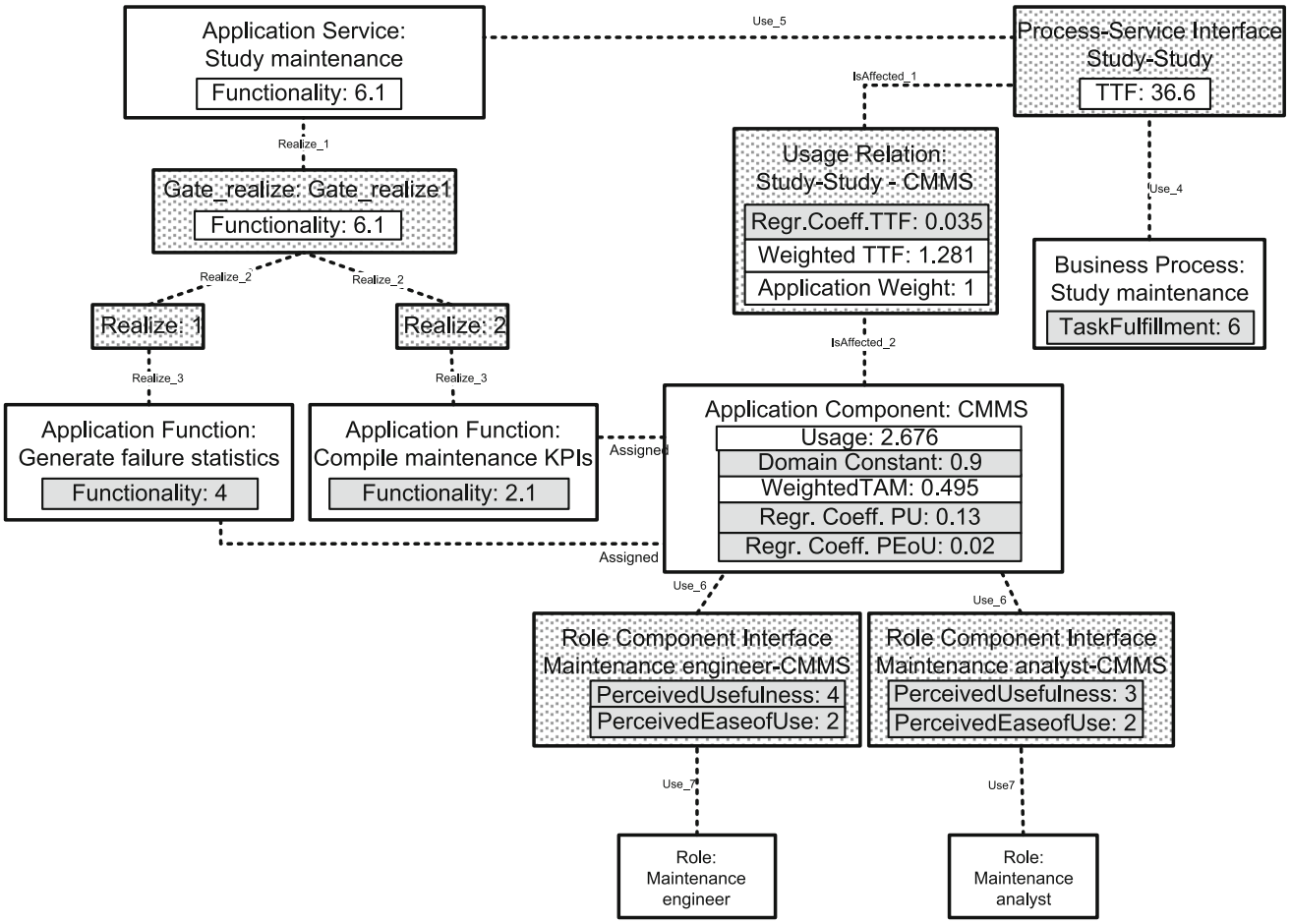


Fig. 4 An example application usage view

Firstly, compile a list of all applications and processes relevant to the inquiry. These lists can be elicited by process managers or anyone with a holistic view of the pertinent processes.

Secondly, perform a survey with a sufficient subset of application users or process performers. For each function of the reference model, the respondents are asked to name the application that implements the function the most and to which degree. For all tasks of the task reference model, the respondents are asked to rate the degree to which they perform the activities. The users are also asked to rate the applications for PU and PEoU.

Thirdly, populate the architecture models with the quantitative data from the surveys and perform the analysis.

An example application usage view To illustrate the use of the viewpoint we present an example view for the fictitious company ACME Energy. The ACME Energy CIO has ordered an exploratory study of the quality of ACME Energy's application portfolio. The application usage viewpoint was employed to determine which applications users liked and would use voluntarily. Here, we model one of the applications, the computerized maintenance management system (CMMS).

In the view of Fig. 4 we see the single `ApplicationComponent` *CMMS* which offers two `ApplicationFunctions` *Generate failure statistics* and *Compile maintenance KPIs* which realize an `ApplicationService` *Study Maintenance* which in turn supports a `BusinessProcess` with the same name. It was discovered that the users considered the CMMS to be all right functionality-wise, which together with a high degree of `BusinessProcess.TaskFulfillment` yielded a high `ProcessService Interface.TTF` for the interface between the *Study* `ApplicationService` and `BusinessProcess`.

Based on the PU and PEoU assessments by the roles `Roles` *Maintenance engineers* and *Maintenance analysts* it was obvious that in spite of the high mark for TTF, the users did not find the *CMMS* to be particularly useful and certainly not easy to use.

To investigate what caused the low PU and PEoU assessments, the architects decided to investigate the service

availability offered by the CMMS. This was done using a dedicated *service availability viewpoint*.

5.2 The service availability viewpoint

The service availability viewpoint is based on the metamodel originally presented in [90].

Concerns The service availability viewpoint addresses the concern of determining the availability of IT services delivered to application users, taking into account both the application and infrastructure layer.

Stakeholder Some likely stakeholders for this viewpoint are service managers and end-users.

Theory Availability is defined as the probability that a service is available to its users [118] over its overall duration of time, which mathematically can be defined as

$$\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \tag{2}$$

where MTTF denotes "mean time to failure" and MTTR "mean time to repair", respectively. MTTF is the inverse of the failure rate (λ) of a component and MTTR is the inverse of the repair rate of a component (μ). The average availability A_{avg} of a component is thus:

$$A_{\text{avg}} = \frac{\mu}{\mu + \lambda} \tag{3}$$

Systems rarely consist of a single component. To model availability in complex systems, three basic cases serve as building blocks; the AND-case where the failure of a single component is enough to bring the system down, the OR-case where a single working component is enough to keep the system up and the *k*-out-of-*n* case in which systems are functioning if at least *k* components are functioning, see Fig. 5. These three cases are used recursively to model more advanced scenarios.

The viewpoint utilizes fault tree analysis (FTA) [112] for the availability analysis.

A first assumption in FTA is independent of failures among different component—implying that there are no common cause failures—which simplifies the modeling task [21].

Furthermore, the assumption of passive redundancy, perfect switching and no repairs is made [48].

When considering components a repaired item is assumed to be in an "as good as new" condition, i.e., assuming perfect repair. If not, assuming a constant MTTF over infinite time will not be valid but instead the component would be in a different state after repair with a different probability of failure. In the ISO 9126-2 standard a similar assumption is stated implicitly [54]. The implications of these assumptions make

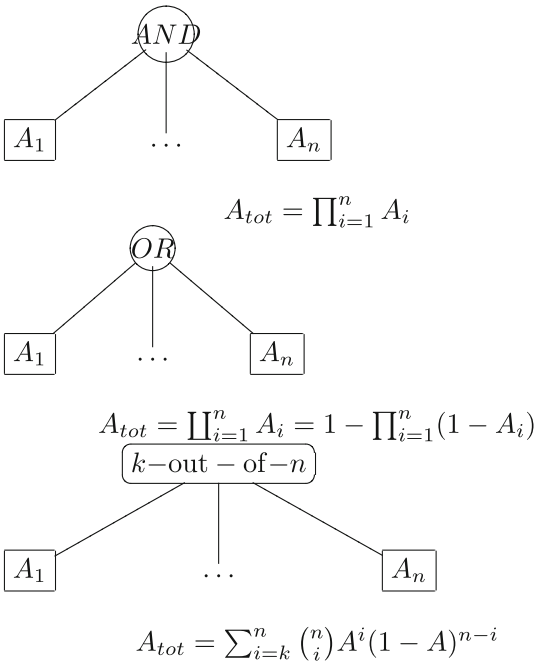


Fig. 5 The basic cases for parallel, serial and *k*-out-of-*n* systems, respectively

it impossible to model common cause failures, active redundancy and a variations in MTTF over time. These assumptions notwithstanding, it was found in [90] that it is possible to make accurate availability assessments.

The availability viewpoint The viewpoint can be found in Fig. 6. The viewpoint incorporates FTA through the introduction of gates which may assume AND or OR characteristics in line with above. The behavior elements are represented by `Services` and `InternalBehaviorElements` (or `Functions` for brief). Both of these have an availability which is represented in the `Service.Availability` and `Function.Availability` attributes, respectively.

`Services` are *realized* by `Functions` and when this is the case, there is a `Gate_Realize` class on the relation between them which qualitatively shows how the realization relation works through the `Gate_Realize.Type` attribute which may assume either one of two states, *AND* and *OR* (to implement *k*-out-of-*n* is left for future works).

Conversely, `Functions` *use* `Services`, and there is a gate on this relation as well: `Gate_Use`.

Acting as an intermediate availability variable on the gates we find the attributes `Gate_Realize.Availability` and `Gate_Use.Availability` and depending on the type of gates, the availabilities are set computed according to Fig. 5, see also Statements 6 and 7.

`Services` are merely externally visible containers of application behaviors and their availability is as such only dependent on the realizing `Functions` and thus identical to the `Gate_Realize.Availability`, see Statement 8.

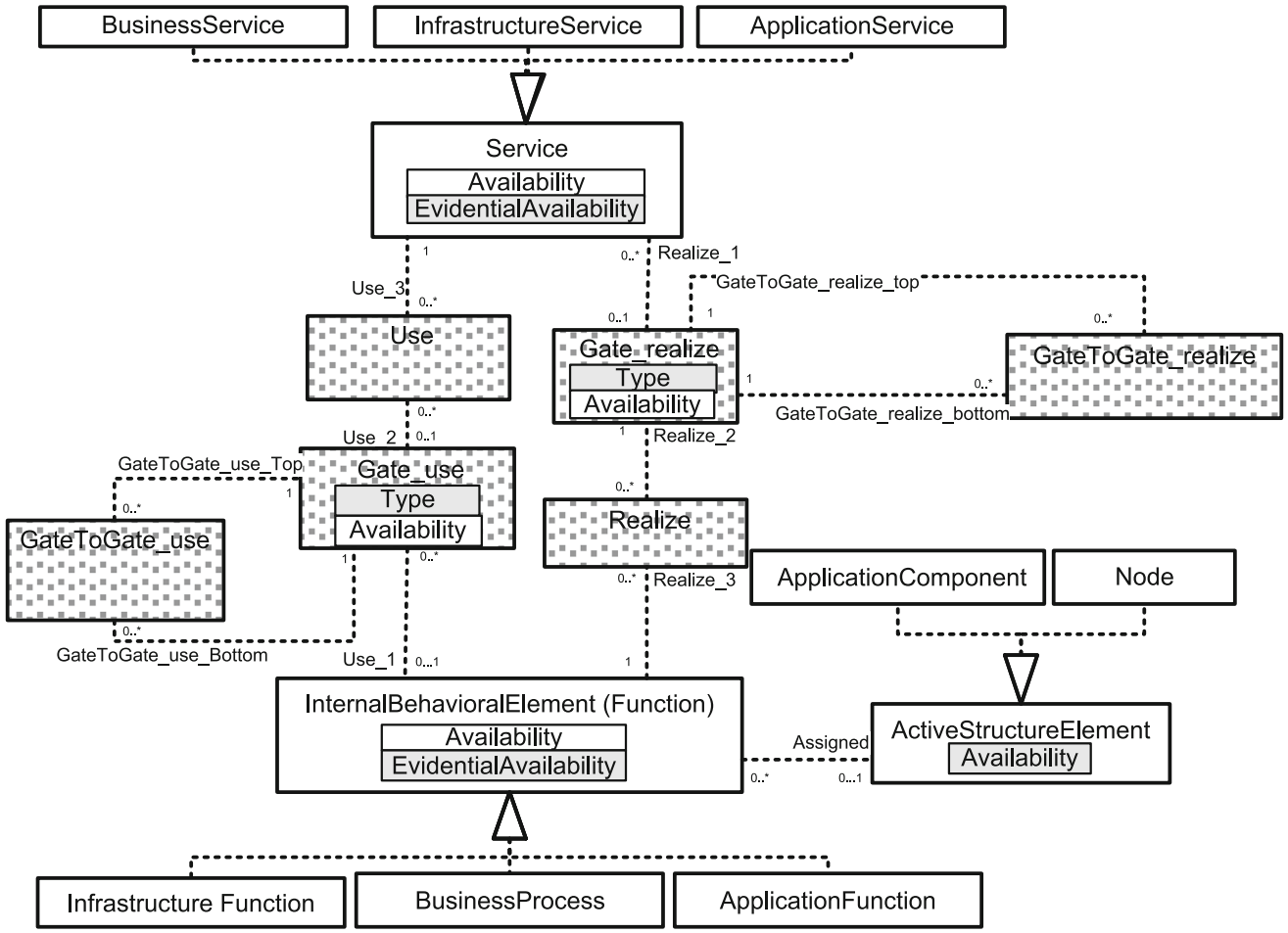


Fig. 6 The service availability viewpoint

`Function.Availability` on the other hand depends also on the `ActiveResourceElement` to which it is assigned. When the `Functions` uses `Services`, `Function.Availability` is the product of the `Gate_Use.Availability` and the `ActiveResource Element.Availability`, since there is an implicit AND relationship between the underlying services and the application realizing the `Functions`, see Statement 9.

Sometimes, there is a need to set the availability directly on a `Function` or `Service`, and this can be done using the attribute `Function.EvidentialAvailability` or `Service.EvidentialAvailability`, respectively.

Guidelines for use The following steps should be taken to use the Service availability viewpoint.

Firstly, identify and scope the service or services of interest, either from a service catalog or through interviews. Defining the service properly is essential to defining what the service being 'available' means.

Secondly, use the viewpoint to qualitatively model the application and infrastructure architecture connected to the service.

Thirdly, elicit quantitative measures of component availabilities. Usually, the easiest way of obtaining the component availability is to ask the respondent (typically a system owner) to estimate how often the component breaks down and estimate the repair time.

Fourthly, run the analysis.

Validation of the service availability viewpoint The viewpoint was tested in five cases with respect to its ability to model and analyze service availability accurately. Furthermore, to investigate the feasibility of the approach, the time spent modeling and analyzing was recorded in each case. Input data was elicited through interviews.

The estimates were compared with existing log files. In each case, the difference in yearly downtimes between the assessed values and the log data was within a few hours of down time per year. Each study required less than 20 man-hours to perform. For the purpose of obtaining good decision support, this indicates that the suggested method yields sufficiently accurate availability estimates.

An example service availability view To probe deeper into the rumors flourishing at ACME Energy regarding the

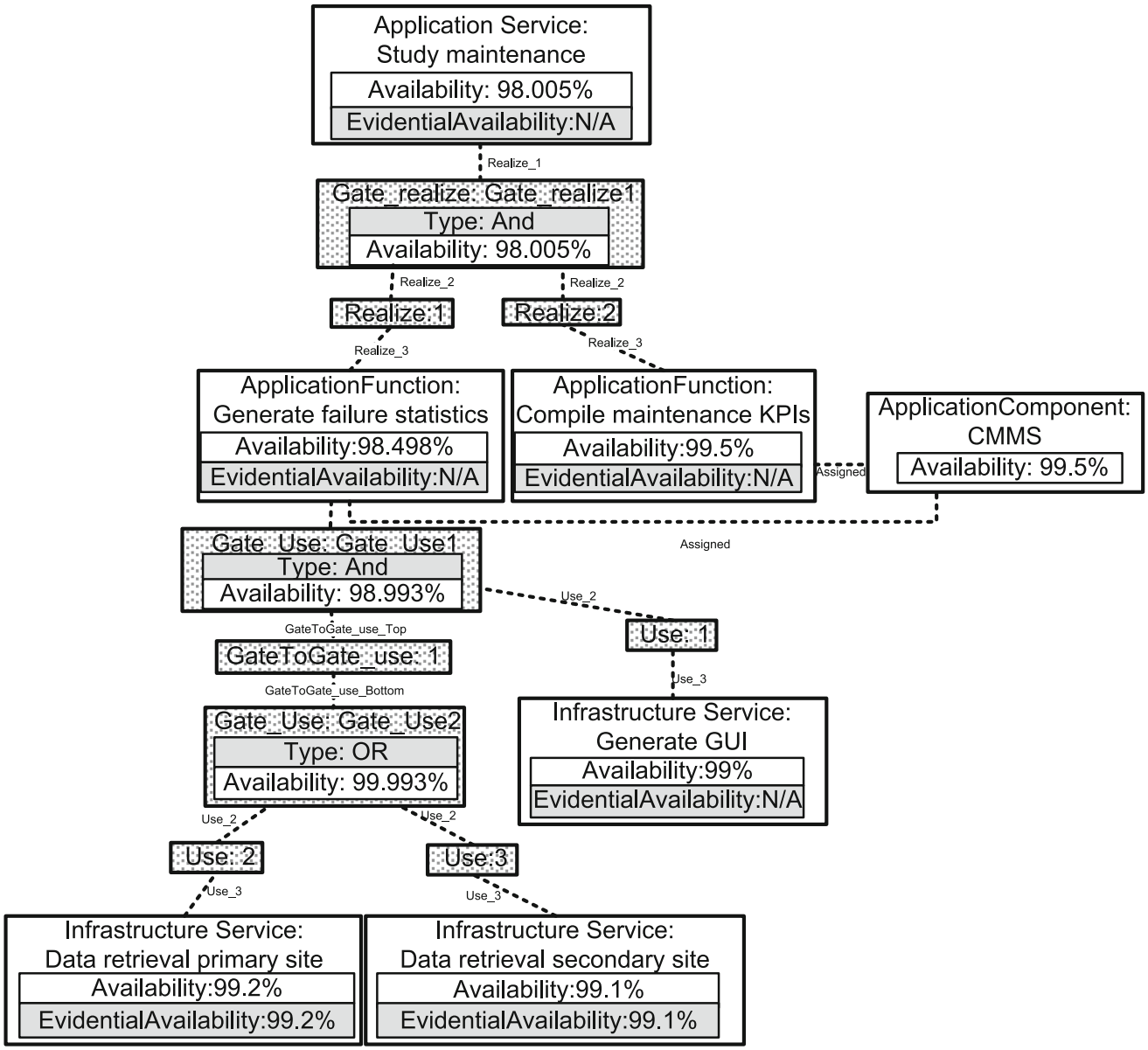


Fig. 7 An example service availability view

incidents affecting the availability of the `Application Service` *Study Maintenance*, the analysts performed an initial round of interviews with system administrators to obtain qualitative data concerning the architecture realizing the `Application Service`. This was modeled according to the service availability viewpoint described above. Quantitative data regarding component availabilities were collected during a second round of interviews.

In Fig. 7 we see the result. The aggregated availability was found to be 98 % which is considered acceptable to most users. Thus, the analysts decide to scrutinize other aspects of the architecture, beginning with service response time.

5.3 The service response time viewpoint

The service response time viewpoint is an adaption of the metamodel which was developed and validated in [85]. That metamodel in turn is an adaptation of the work done by Iacob and Jonkers [49].

Concerns The viewpoint is used to analyze service response time taking both the application and infrastructure layer into account.

Stakeholders Service managers interested in maintaining agreed service levels are obvious stakeholders, but also end-user organizations wishing to ascertain that changes to

the present architecture will result in acceptable service levels.

Theory This section will introduce a service response time analysis framework employing common queuing networks as presented by [50]. To use the framework, the analyst needs to model structural elements, the internal behavior elements they offer and the externally exposed services.

The approach for workload estimation is top-down and begins with the arrival frequency of usage requests from the business layer which is converted into arrival rates for the underlying components in the architecture. Based on the workload, response times of the behavior components and utilizations of the resources can be determined bottom-up.

The arrival rate λ_a of behavioral node a (referring to anyone of the behavioral concepts in ArchiMate) is computed using Eq. 4. Here d_a^+ is the number of outgoing relations to other components, i.e., components that use or are realized by component a. k_i refers to one of the d_a^+ child components of component a, i.e., those that use or are realized by component a. λ_{k_i} refers to the child components' respective arrival rates. n_{a,k_i} is the number of times node a is used by component k_i. f_a is the local arrival frequency of component a. Local frequency refers to arrival rates which are incurred on node a from other parts of the structure not modeled in the architecture model.

$$\lambda_a = f_a + \sum_{i=1}^{d_a^+} n_{a,k_i} * \lambda_{k_i} \tag{4}$$

The utilization of resource r, U_r refers to the faction of the resource that is being used is found recursively using Eq. (5). Here, C_r refers to capacity, which in this context means the number of servers.

$$U_r = \frac{\sum_{i=1}^{d_r} \lambda_{k_i} * T_{k_i}}{C_r} \tag{5}$$

where d_r is the number of behavior components k_i which are assigned to the resource, λ_{k_i} is the arrival rate and T_{k_i} is process time, which is computed as follows:

$$T_a = S_a + \sum_{i=1}^{d_a^-} n_{k_i,a} * R_{k_i} \tag{6}$$

where d_a^- denotes the "in-degree" of node a, i.e., the number of parent components of component a that are either used by component a or realizing component a. k_i is a parent of a, r_a is a resource assigned to a and R_{k_i} is the response time of a, to which we will return below. The internal service time S_a is taken to be a known constant for every behavior element.

To compute the response time, there is a need to determine which queueing model to use. One of the most common is the $M/M/1$ model, which assumes Poisson distributed arrival rates, that the Service time is exponential and that a single server queue [57]. Under these assumptions R_a becomes

$$R_a = F(a, r_a) = \frac{T_a}{1 - U_{r_a}} \tag{7}$$

Other models include the $M/M/s$ model with multiple servers (i.e., when $C > 1$), or the $M/G/1$ model which assumes Poisson distributed arrival rates, but no knowledge of service time distributions [46]. Queues to the resources are treated as independent from each other which will introduce minor errors in the performance estimates [49].

The service response time viewpoint Fig. 8 shows the service response time viewpoint. `Services` are *realized* by various kinds of `InternalBehaviorElements`, (as usual denoted `Functions` for brevity). From a response time perspective, the realization relation in itself does not unambiguously state how `Functions` realize `Services`. In the original work by [49], it was assumed that there was a one-to-one relation between `Functions` and `Services`, and that each `Function` is called upon only when realizing a `Service`.

However, when integrating the service response time viewpoint with the other viewpoint, the one-to-one assumption does not hold; to be able to integrate with the Service availability and application usage viewpoints, the present viewpoint must allow many-to-many relations between `Functions` and `Services`. Therefore, there is a need to introduce the class `Gate_Realize` between `Functions` and `Services`. This class has the attribute `Gate_Realize.ExecutionPattern` which may assume the values *(Serial, Parallel)*. The former value means that the `Functions` are used sequentially, and the latter that they are used in parallel, this is very similar to the approach for response time aggregation suggested in [56].

Depending on the execution pattern, the response time propagations between the `Functions` and `Services` will differ. The gate class also contains the attribute `Gate_Realize.ResponseTime` which is an intermediate response time which is

$$Gate\_Realize.ResponseTime = \sum_{i=1}^{n} R_i \tag{8}$$

in the serial case, or

$$Gate\_Realize.ResponseTime = \max R_i \tag{9}$$

in the parallel case. R_i is the response time of `Function` i realizing the `Service`, see Statement 10.

Since it may be the case that several `Functions` are used when realizing a `Service`, the viewpoint expresses this in an attribute (used in Eqs. 4, 6) belonging to the placeholder class `Realize` attribute `Realize.Weight`. Following Eq. 4, we introduce the `Realize.WeightedWorkload`

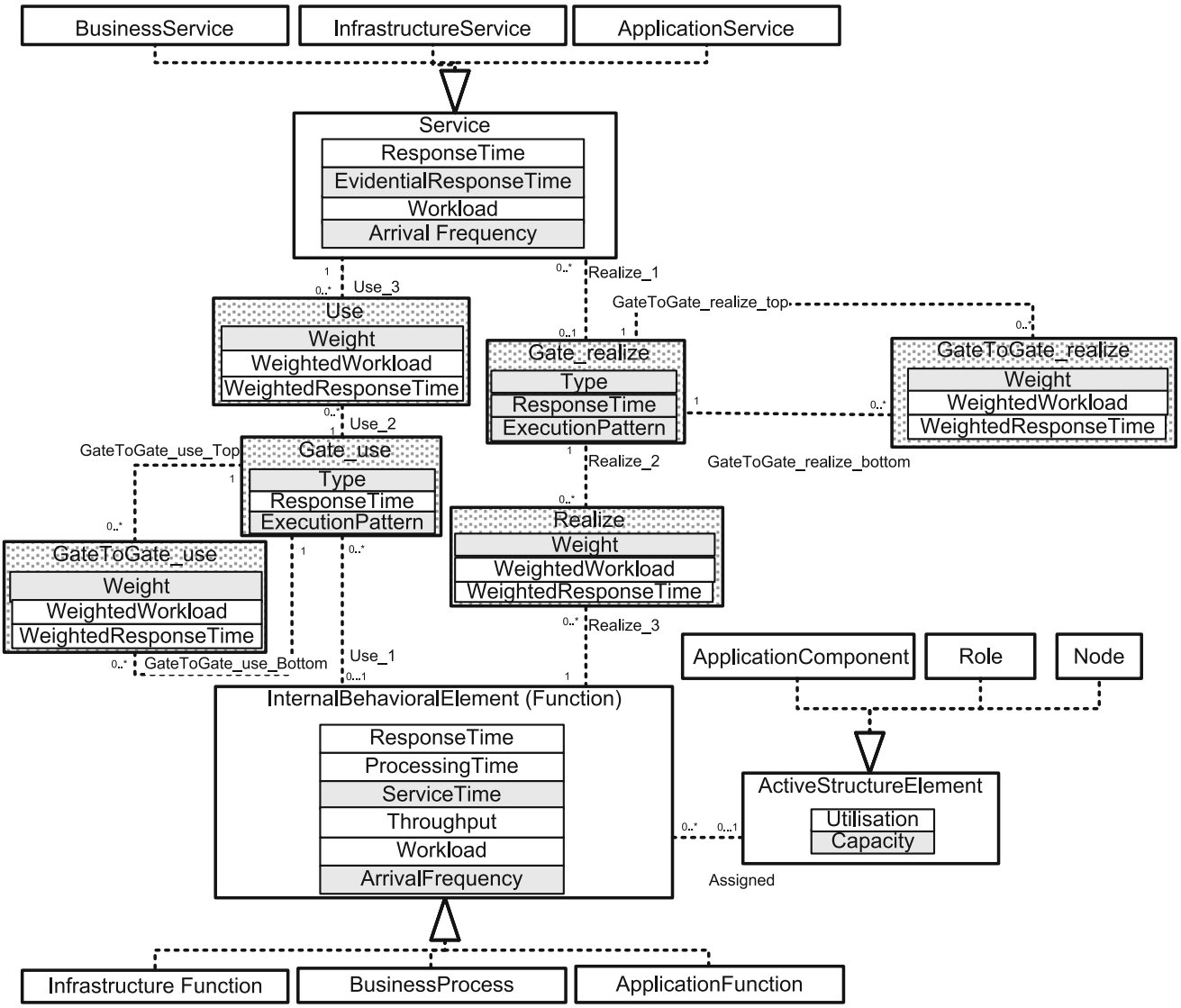


Fig. 8 The service response time viewpoint

attribute, see Statement 12. To be able to accommodate multiplying the weight with the response time from underlying nodes as defined in Eq. 6, the attribute `Realize.WeightedResponseTime` is also introduced, see Statement 11.

The `Gate_Realize` class can be connected to other `Gate_Realize` classes through the `GateToGate_Realize` class which has weight, weighted workloads and response time attributes. This allows the modeler to model arbitrarily complex compositions of `Functions` and to be able to recursively propagate the workload from `Services` to `Functions` using virtually the same expression as Statement 12.

`Service.ResponseTime` is identical to the response time of the its closest `GateRealize` class, see Statement 13. `Service.Workload` will depend on both the arrival frequency (i.e., the frequency of invocations from business processes or other services not shown in the model) and the frequency of calls from `Functions` which *use* the `Service`. The attribute `Use.WeightedWorkload` of the class closest to the `Service` will determine the `Service.Workload`, see Statement 14.

`Functions` are *assigned* to `ActiveStructure-Elements` and use `Services`. The workload of the functions is derived from the `Services` realized by the `Function`, and by the arrival frequency of requests from `Services` not explicitly modeled. Thus, the attribute `Function.Workload` is determined by the sum of the attributes `Function.ArrivalFrequency` and `Realize.WeightedWorkload` for all `Realize` objects which are directly related to the `Function`, see Statement 15.

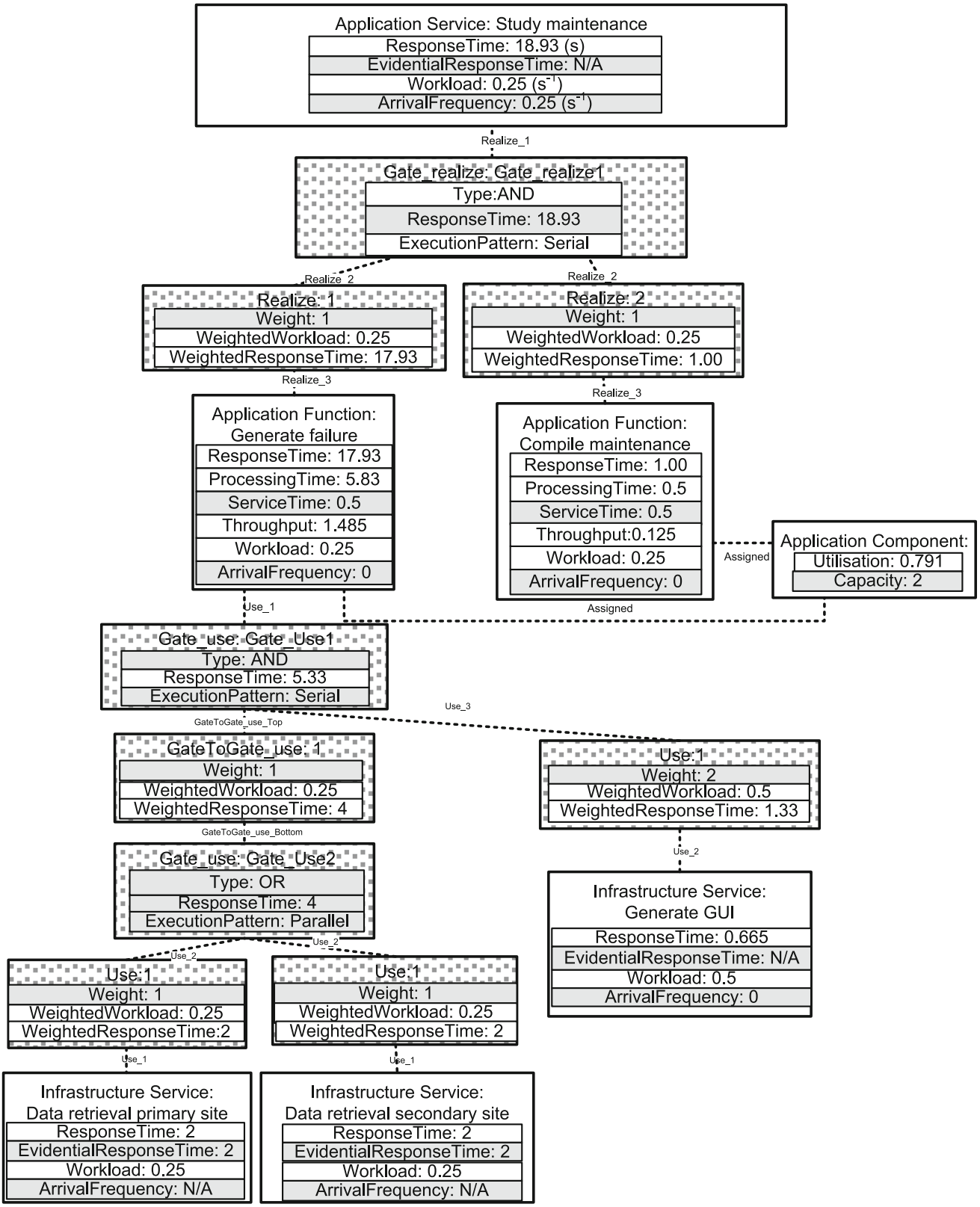


Fig. 9 An example service response time view

To calculate Function.ResponseTime there is a need to know the Function.ProcessingTime, which is the sum of the internal Function.ServiceTime and the Service.ResponseTimes of all used Services, see Statement 17.

The ActiveStructureElement to which a Function is assigned has some ActiveStructureElement.Utilization. To find this value, we introduce the attribute Function.Throughput; the product of Function.Workload and Function.ProcessingTime, see Statement 16. ActiveStructureElement.Utilization is the quotient of Function.Throughput and ActiveStructureElement.Capacity, where the latter refers to the number of identical ActiveStructureElements to which the Functions are assigned, see Statement 18.

Depending on the kind of queue Function.ResponseTime will be computed differently, see e.g., Eq. 7. The present implementation allows for $M/M/1$ queues (when ActiveStructureElement.Capacity is one) and $M/M/s$ (when ActiveStructureElement.Capacity exceeds one), see Statement 19 which implements both of these following [46].

Validation A case study was conducted at a Swedish power company where a total of five ApplicationServices were evaluated using the modeling and analysis method described above. Input data came from interviews with system experts and a survey with application users to determine the service workload. The results of these evaluations were compared with measurements of the response times of said Application Services and the differences were within 15 % for four out of five services. Furthermore, using the proposed viewpoint consumed a third of the time it took to measure the same values using an experimental approach which leads us to believe that it is a resource-efficient and fairly accurate method.

Guidelines for use The following steps should be taken when employing this viewpoint.

Firstly, select and scope a service to be measured.

Secondly, perform the qualitative modeling and model all relevant objects that are connected to the service. The respondent or respondents need to be knowledgeable about the system architecture.

Thirdly, elicit workload data. If the number of users is large, a survey might be used, else use interviews.

Fourthly, elicit the the remaining component performance parameters. The respondents could, e.g., be system administrators.

Fifthly, run the analysis.

An example service response time view The analysts suspect that the Study Maintenance ApplicationService might be insufficient with respect to service time and decide to use the service time viewpoint described here to investigate this further. Users indicated that at the end of contracting periods when maintenance engineers from all of ACME Energy perform contractor evaluations, the workload is quite high for the Study Maintenance ApplicationService, which sometimes results in high response times. The analysts therefore model ed the scenario of peak load for the service.

Figure 9 is a service response time view of the Study Maintenance ApplicationService. We see that the response time for of the Application Service Study Maintenance is around 19 s, which is fairly high but acceptable to most users. However, the Application Component.Utilisation of the CMMS is quite high, and could thus be a possible future bottleneck. In summary, service time does not appear to be the reason why users do not like the CMMS and the analysts decide to focus more on data accuracy.

5.4 The data accuracy viewpoint

This subsection describes the data accuracy viewpoint, which is an adaptation of the metamodel from [87].

Concerns Using this viewpoint makes it possible to estimate the accuracy of data sets within the organization. It is also possible to determine which applications or business process that introduce errors into the data sets.

Stakeholders Obvious stakeholders are data custodians, i.e., those in charge of maintaining data quality, but also end users wishing to know the quality of the data which they use in their daily activities.

Theory The present viewpoint employs process modeling in a manner similar to that of IP maps and that of [4]. Furthermore, following [22], the viewpoint also shows how data can improve when manipulated in business processes.

The Passive Component Set is used to describe sets of information objects whether stored in databases (then specialized into DataSets) or as more unstructured information (specialized into RepresentationSet). The attribute PassiveComponentSet.Accuracy is defined below.

Firstly, we denote the individual Representations and DataObjects PassiveComponentObjects. Next, we introduce the following:

N: Number of PassiveComponentObjects in the PassiveComponentSet
N^{acc}: Number of accurate PassiveComponentObjects in the PassiveComponentSet

N^{inacc}: Number of inaccurate `PassiveComponentObjects` in the `PassiveComponentSet`

where "accurate" or "inaccurate" for the `PassiveComponentObjects` is defined as their value V being sufficiently close to the true value V' in line with [6,98].

Since `PassiveComponentObjects` can be either accurate or inaccurate we have

$$N^{\text{acc}} + N^{\text{inacc}} = N. \tag{10}$$

The accuracy of the `PassiveComponentSet` can then be defined as

$$PassiveComponentSet.Accuracy = \frac{N^{\text{acc}}}{N} \tag{11}$$

The number of accurate `PassiveComponentObjects` in a `PassiveComponentSet` may change when processed by a `Function` or a `Service`. These may corrupt a `PassiveComponentObject` which was accurate at process step $T = t$ into being inaccurate at time step $T = t+1$. To be able to reason about this we introduce N^{det}: the number of accurate `PassiveComponentObejects` at process step $T = t$ which were made inaccurate by a `Function` or a `Service` at process step $T = t + 1$.

The frequency of this happening is

$$\alpha = \frac{N^{\text{det}}}{N_t^{\text{acc}}} \tag{12}$$

Similarly, an `Function` or a `Service` may correct inaccurate `PassiveComponentObjects`. We introduce N^{corr}: the number of `PassiveComponentObjects` that were inaccurate at process step $T = t$ but made accurate by a `Function` or a `Service` at time step $T = t + 1$.

$$\beta = \frac{N^{\text{corr}}}{N_t^{\text{inacc}}} \tag{13}$$

The number of accurate objects at process step $T = t + 1$ is given by

$$N_{t+1}^{\text{acc}} = N_t^{\text{acc}} - N^{\text{det}} + N^{\text{corr}} \tag{14}$$

From the above, an expression of the accuracy of a `PassiveComponentSet` at $T = t + 1$ can be derived:

$$\begin{aligned} PassiveComponentSet.Accuracy_{t+1} &= \frac{N_{t+1}^{\text{acc}}}{N} \\ &= \frac{N_t^{\text{acc}}}{N} - \frac{N^{\text{det}}}{N} + \frac{N^{\text{corr}}}{N} \\ &= \frac{N_t^{\text{acc}}}{N} - \frac{\alpha * N_t^{\text{acc}}}{N} + \frac{\beta * N_t^{\text{inacc}}}{N} \\ &= \frac{N_t^{\text{acc}}}{N} * (1-\alpha) + \frac{\beta(N - N_t^{\text{acc}})}{N} \\ &= \frac{N_t^{\text{acc}}}{N}(1-\alpha) + \beta(1 - \frac{(N_t^{\text{acc}})}{N}) \\ &= PassiveComponentSet.Accuracy_t * (1-\alpha) \\ &\quad + \beta * (1 - PassiveComponentSet.Accuracy_t) \end{aligned} \tag{15}$$

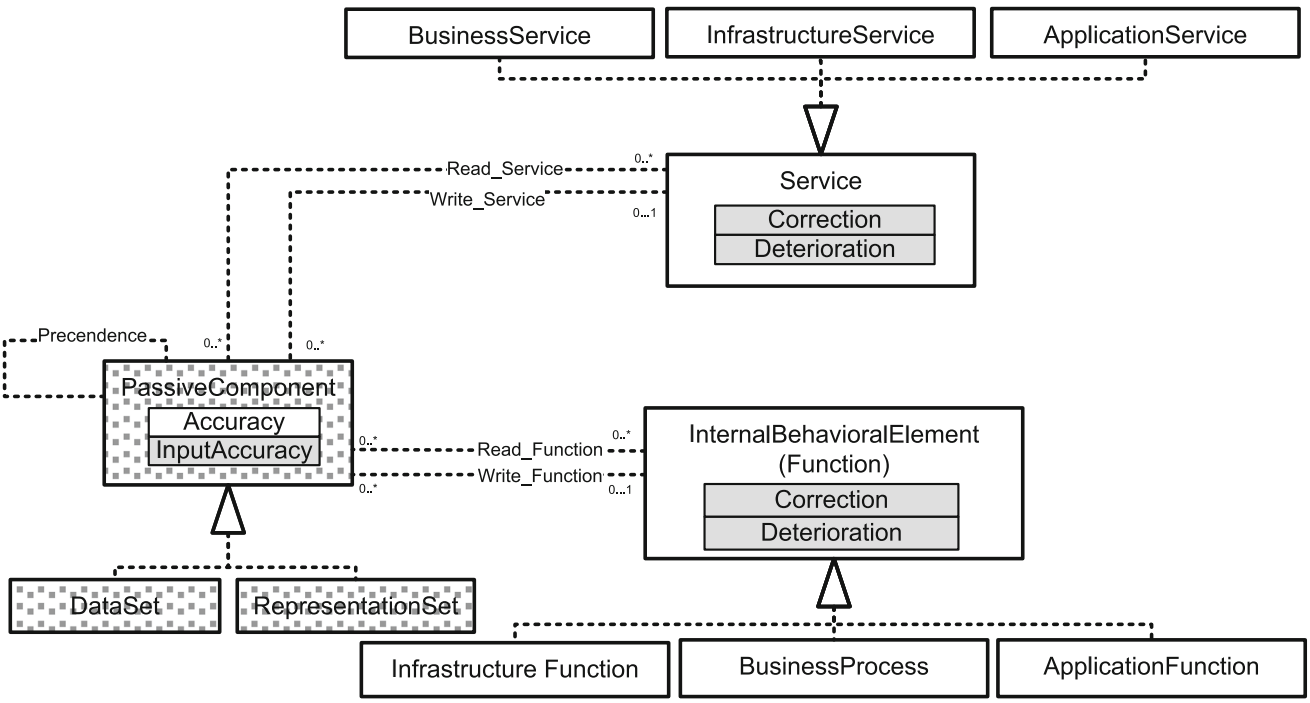


Fig. 10 The data accuracy viewpoint

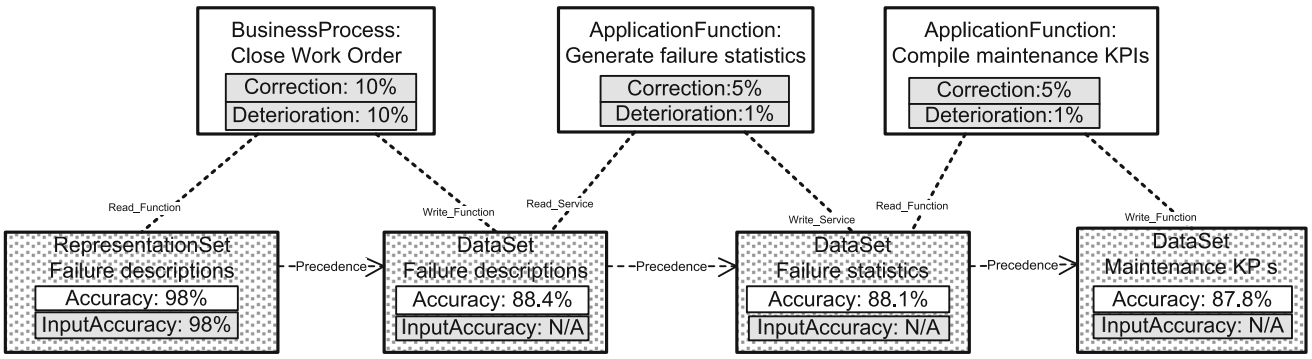


Fig. 11 An example data accuracy view

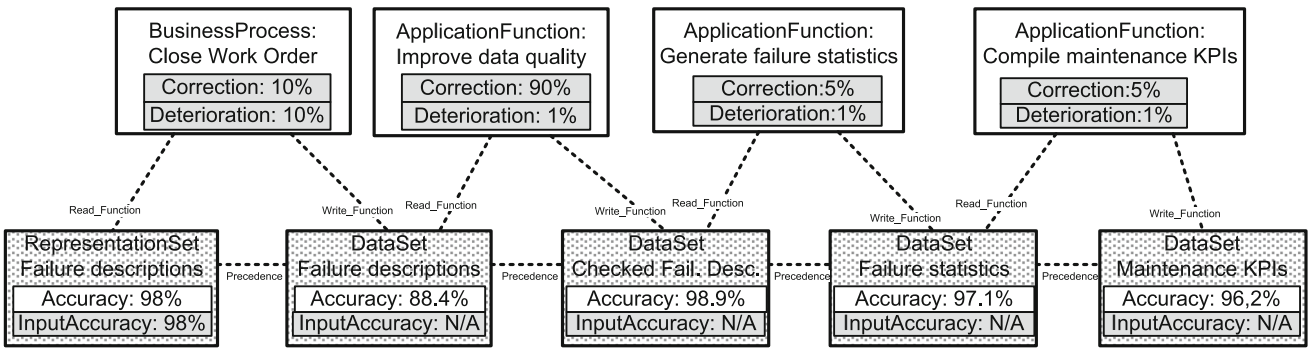


Fig. 12 The data accuracy view for the suggested to-be scenario

The data accuracy viewpoint The data accuracy viewpoint can be found in Fig. 10.

The properties α and β are found as attributes `Function.Correction`, `Function.Deterioration`, `Service.Deterioration` and `Service.Deterioration`. Whenever a `PassiveComponentSet` is read or written by a `Service` or `Function` these attributes either improve or deteriorate the `PassiveComponentSet.Accuracy`, see Statement 20. `PassiveComponentSet.InputAccuracy` is an attribute used to specify the baseline accuracy of the first `PassiveComponentSet` in the process.

Validation The data accuracy viewpoint was tested in a case study at the same Swedish power company in which the service response time was tested, see [87]. Using interviews and the viewpoint a process model was created and the accuracy of a `DataSet` was estimated to 94.905 %. When sampling 37 `Data Objects` from the same `DataSet`, the accuracy was determined to be 94.6 %, a rather small difference suggesting that the viewpoint is indeed useful. It took significantly longer to do the sampling (17 man-hours) than the modeling and analysis (11 h) which indicates that the present viewpoint is resource-efficient to use.

Guidelines for use To use the viewpoint follow the following process.

Firstly, model the data flow qualitatively. Suitable respondents are those performing the process who understand the process side of the flow or system architects which understand the application side of things.

Secondly, elicit parameters input accuracy, deterioration and correction from the same respondents.

Thirdly, run the analysis.

An example data accuracy view An example data accuracy view can be found in Fig. 11. The ACME Energy analysts decide to investigate whether the reason CMMS users hold the application to a low esteem is due to a poor data accuracy in the information provided by the application.

One important piece of information used when compiling the maintenance key performance indicators (KPIs) is the field "Failure description" which the maintenance workers use to report what caused a failure in a piece of equipment. This is reported as a part of closing the work order which was issued when the failure was first detected. Eliciting estimates of the correction and deterioration attributes as well as the input accuracy of the processes and services was done through interviews. Using these estimates and the viewpoint above, it was estimated that the accuracy of the output *Maintenance KPIs* (with respect to failure statistics) was 87.8 %. This is a low number, and in order to improve the perceived usefulness of the application improving this number might be a viable option.

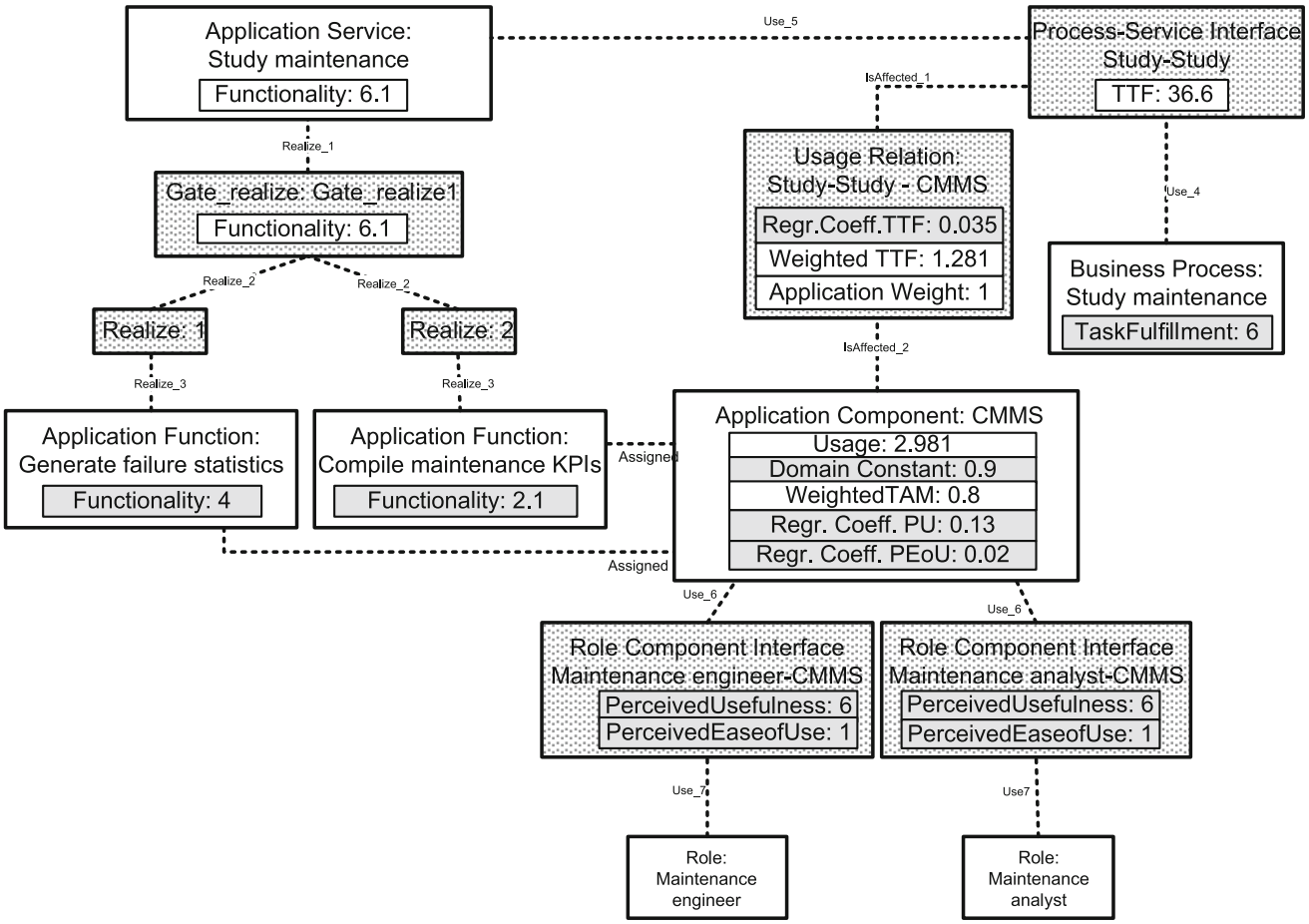


Fig. 13 The application usage view for the suggested to-be scenario

5.5 Decision-making concerning future architecture changes

On behalf of the ACME CIO, the architects were given the task to find a way of improving the poor data accuracy: to implement an `Application Function` with which to perform automatic quality checks of data consistency and accuracy when closing the work order. See Fig. 12. It was decided based on some initial tests that such quality checks would probably correct 90 % of errors, while deteriorating approximately 1 % of the data objects in the set. From Fig. 12 it is evident that he introduction of these checks would enhance output data quality significantly, from the original 87.8 to 96.2 %.

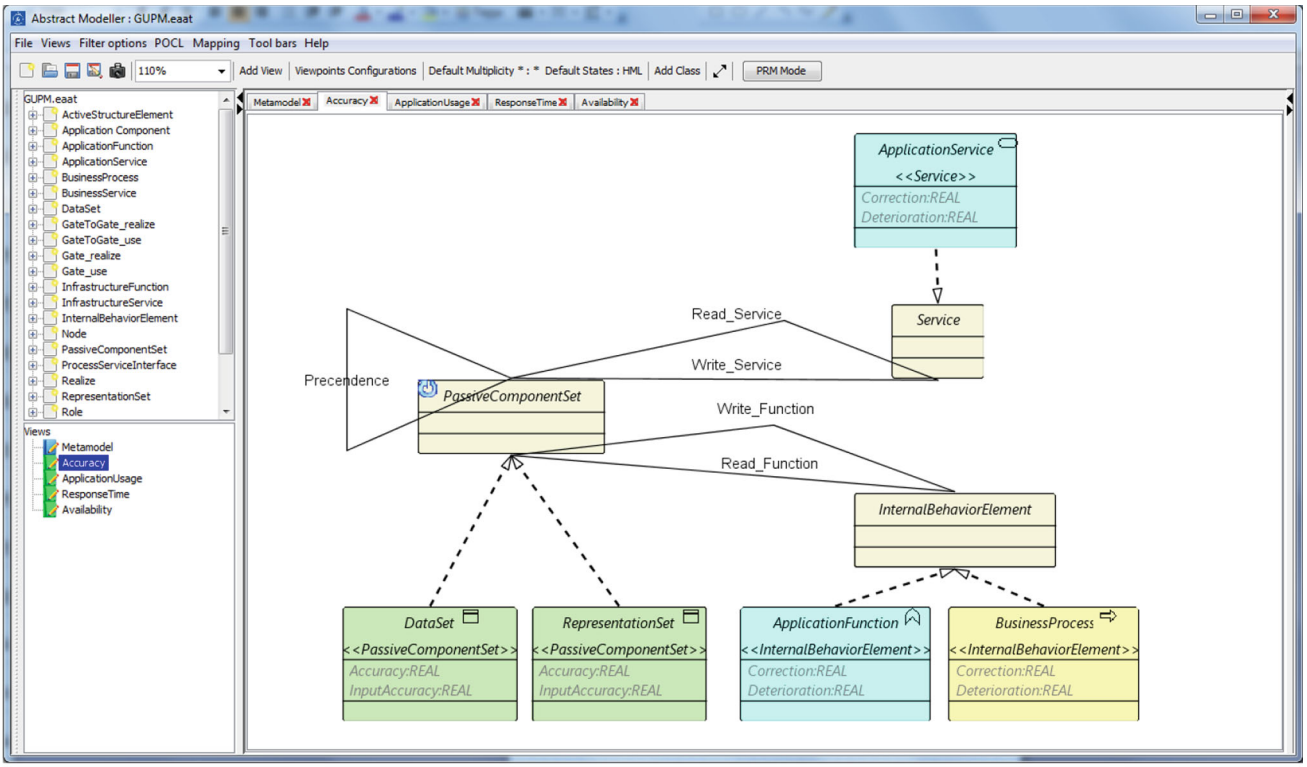
The impact on performance and availability was found to be negligible, but the impact on application usage was not. By making a user survey with test users of the pilot data quality enhancement implementation it was concluded that users found the new interface more difficult to use while at the same time greatly appreciating the increase in data accuracy. Quantitatively, this translated into a decrease in perceived ease of use for both involved roles and an increase in perceived usefulness.

To handle the trade-off between the increase in usefulness and the decrease in ease of use the scenario was modeled using the Application usage viewpoint. The predicted application usage rose from about 2.68 to 2.98 which is an 11 % improvement over the present situation, see Fig. 13. ACME's application architect therefore recommended to the CIO to include the function in the next release of the CMMS software.

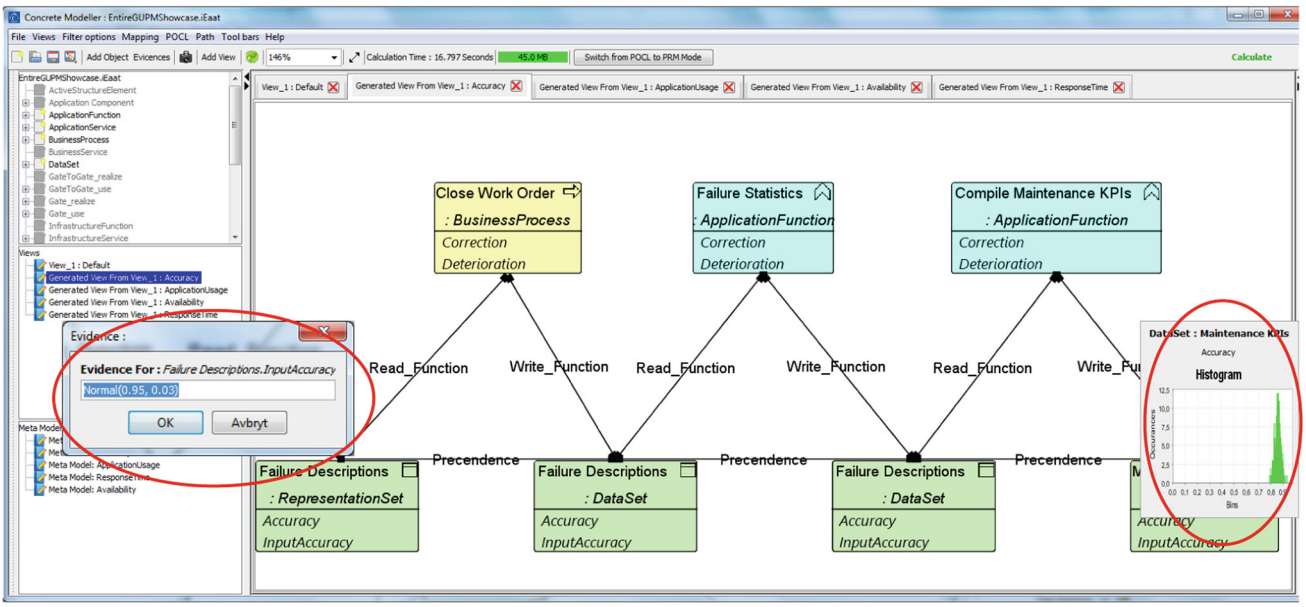
5.6 Tool implementation

As noted in Sect. 3, the present framework has been implemented in the EAAT tool. Below are screenshots of the accuracy viewpoint (Fig. 14a) and the accuracy view (Fig. 14b) as implemented in the tool.

Since the p-OCL formalism is probabilistic, it is possible to insert uncertain evidence. For instance, a respondent might not know the exact value of the input accuracy but instead approximate it with a normal distribution with a mean of



(a) The accuracy viewpoint in the EAAT tool



(b) The accuracy viewpoint in the EAAT tool

Fig. 14 EAAT Tool screenshots: **a** the accuracy viewpoint and **b** the accuracy view with some probabilistic evidence inserted

95 % with a variance of 0.03 (see the little box 'evidence' to the left in Fig. 14b). Using the Monte Carlo simulations of the tool, a select number of iterations can be performed to reach the final output value which is normally distributed as well, see the box with the histogram to the right in Fig. 14b. Thus, decision-makers using the tool may be able to judge the degree of uncertainty in the architecture analysis.

The implementation of the metamodel and the viewpoints of this paper are available for download, see Appendix C for further details.

6 Discussion

6.1 Findings

The present paper presents an EA framework featuring four viewpoints addressing different concerns. In being concerned with modeling enterprise information systems and their business contexts the framework can be considered an IT artifact [45]. As such it belongs in the research stream commonly referred to as design science [3,42,45,78,91,123].

This section will elaborate on the qualities of the work presented through framing and comparing it with the criteria from [42], which state that a design science theory should comprise eight structural components:

1. Purpose and scope—what purpose(s) does the theory fill and what are the limit(s) to its use?
2. Constructs—which are the basic constructs involved in employing the theory?
3. Principles of form and function—how does the artifact behave?
4. Artifact mutability—how does the artifact vary with its environment?
5. Testable propositions—which are the theory's testable propositions?
6. Justificatory knowledge—How do we justify stating that the theory works?
7. Principles of implementation—Which are the principles of implementations for practitioners?
8. Expository instantiation—Is there an instantiation to further understanding of the artifact?

Purpose and scope The purpose of the present EA framework can be summarized in a number of 'meta requirements' [123]. The artifact aids modeling of enterprise architectures comprising both information systems as well as parts of the business environment so as to make the models amenable to analysis of four properties: the likelihood of application usage, the availability of services, the response time services and data accuracy.

Another meta-requirement is that the modeling and analysis should be resource-efficient even when data are scarce: the artifact does not pre-suppose that there are architecture models available, nor that the organizations employing the artifact need to procure expensive equipment such as availability logging equipment to measure the service availability. This has led to the use of data collection methods based on surveys and interviews, which have been demonstrated to be resource-efficient elsewhere.

Constructs The constructs of the present EA framework are primarily the classes from the ArchiMate metamodel. These have been augmented with a number of classes which are needed to support the analysis as well as attributes that capture a number of variables. Relations between the attributes themselves, and between classes and attributes are described in p-OCL-statements found in Appendix 7.

Principles of form and function The overall principles of how the EA framework behaves have been sketched for each viewpoint in Sect. 5. Additionally, the exact analysis mechanisms of the models have been detailed in the p-OCL statements of Appendix 7.

Artifact mutability The present EA framework must be adapted to fit its environment. This is particularly so for the application usage viewpoint which requires functional and process descriptions to fit the application domain. For service response time it is possible to expand the viewpoint to also encompass business services. The response time calculations are contingent upon assumptions made about arrival rate and service time distributions (Poisson and exponential, respectively); under other assumptions other queuing models would apply. The availability equations hold under the assumptions of exponential failure rates and it is conceivable that other distributions, for instance the log normal, should be used in some situations.

Testable propositions Each viewpoint implies a testable proposition of the form, "is it possible to yield accurate availability/response time/accuracy/application usage predictions using viewpoint X and input data collection method Y", where X is one of the four viewpoints and Y is either interviews or surveys. These have been tested elsewhere in [85–87,90]. A brief summary of the results from these studies are presented in Table 1, they demonstrate that employing the metamodels together with the suggested data collection methods yields fairly accurate results.

Furthermore, in the cases data accuracy, service availability and service response time, the effort required to perform the case studies and analysis was recorded, see Table 2. In these cases, the researchers began from scratch with creating the models and collecting the data. When applying the integrated approach, the effort spent per property is likely to fall substantially since it is possible to re-use the architecture content for multiple analyses. Thus, the model should be useful in a practical setting.

As for the overall framework presented here, the testable proposition is 'is it possible to integrate the four metamodels into one integrated metamodel and still retain the analysis capabilities of the individual metamodels'. This has been tested through example instantiations where a number of views have been implemented. More importantly, the entire framework has been successfully implemented in the EAAT tool, with the analysis capabilities intact.

Justificatory knowledge The viewpoints are based on sound and previously published 'kernel theories' [123].

Table 1 Summary of the validation activities

Property	Type	Data collection	No. respondents or cases	Results
Application usage [86]	User survey	Online questionnaire	55 respondents	The model was able to explain variations in user behavior
Service availability [90]	Case study	Interviews/ observations	5 organizations, 7 services	The availability predictions were all within a few hours of the actual, measured yearly downtimes
Service response time [85]	Case study	Interviews/ experiments	One organization, 5 services	4 out of 5 predictions within 15 % of measured service response times
Data accuracy [87]	Case study	Interviews/ observations	One organization, one data set	Predicted accuracy within 0.1 % of actual

Table 2 The effort spent doing the case studies for three of the property assessments

Property	Effort case study (man hours)
Service availability	<20
Service response time	<20
Data accuracy	<12

The effort takes both the researcher's and the organization's time into account

For service availability, the underlying kernel theory is fault tree analysis, which is commonly employed by practitioners for reliability and availability analysis of complex systems. For service response time the underlying kernel theory is queuing theory which is employed extensively for performance analysis previously. For data accuracy, the work of Cushing [22] and Ballou et al. [4] constitute kernel theories albeit with modifications. For application usage, the technology acceptance model (TAM) [23] and the task-technology fit (TTF) model [41] serve as kernel theories.

Principles of implementation Each viewpoint description includes basic method guidelines for users (who are most likely enterprise architects). Since the viewpoints are also implemented in the EAAT tool, which is free for anyone to download and use, practitioners can easily start using the framework from this paper.

Expository instantiation The integrated and revised metamodel has been instantiated in four example views in this paper. Furthermore, screenshots from the implemented models in the tool EAAT have been shown, to illustrate what the models look like in a tool implementation.

6.2 Limitations and future works

An obvious limitation with the current framework is that it comprises four viewpoints only. In the non-functional property sphere alone there are several other concerns which could be addressed using architecture models, e.g., security [111], interoperability [116] or modifiability [69]. An fruitful next step would be to integrate these architecture metamodels as viewpoints into the current framework.

So far, ArchiMate has served as the basis for the metamodel, an interesting future work could be to test the method using other architecture metamodels as foundations as well.

To further enhance the predictive capabilities of the application usage viewpoint it is possible to use a kernel theory that provides even more explanatory power. The unified theory of acceptance and use of technology [120] could be used as a starting point in this respect.

The proposition that EA aids decision-making is commonly encountered [60,61,68], but with the exception of [38], there is little research done on actually employing the EA frameworks to aid decisions. A case study involving the current framework, a decision concerning the EA and some way of evaluating decision quality would be very valuable.

The viewpoints have been tested individually using their original metamodels, but mostly in very few cases: both service response time and data accuracy in a single case study, and application usage for one application domain only. More studies are thus needed to test the individual properties. Furthermore, the overall framework presented here has made some changes to the original metamodels and apart from an example instantiations this new integrated framework has not been tested in its entirety, which is left to future works.

6.3 Contributions

The above limitations aside, the present framework must still be seen as a valuable artifact. To practitioners it offers both method and modeling assistance in evaluating several properties which are in themselves closely related to achieving net benefits from information systems [26].

To researchers, the present framework offers a foundation on which to either integrate additional viewpoints to extend the analysis capabilities, for instance by adding security or interoperability analyses.

To researchers within enterprise modeling, the present framework offers some input into which constructs are of use when modeling for architecture analysis. The extensions that have been made to ArchiMate indicate areas of improvement: by including `Gates` for the fault tree based availability analysis, by adding classes on the `Realized` and the `Used` relations to be able to express weights for the response time case, by adding the `ProcessServiceInterface` class between the business process and the application services to be able to state something about the matching of the functionality with the task requirements, to add the class `RoleComponentInterface` between the `Role` class and the `Application Component` class thus being able to capture user opinions of the application components. Furthermore, the addition of attributes could serve as input to the ArchiMate work.

7 Conclusions

This article describes an EA framework which can be employed for modeling and analysis of four properties, viz.: (i) application usage, (ii) service response time, (iii) service availability, and (iv) data accuracy. The present work integrates metamodels presented in previous work and presents them as four viewpoints with brief introductions to their underlying theory and short accounts of their validation and testing. The instantiation of these viewpoints into views are shown by means of a running example. p-OCL statements describing the exact analysis mechanisms are also provided. The measures can be used to either assess the as-is architecture to explore which parts of it to make changes to, or they can be applied to future scenarios thereby making them comparable to the decision-maker.

Appendix A: p-OCL statements

Below are the attribute definitions and operations employed for this paper. In Figs. 3, 6, 8 and 10, we have omitted the role labels on the relations to avoid cluttering the figures to much. In the p-OCL statements below, it is assumed that the 'main direction role' gets the same name as the relation name, but with a small case letter. The other role has the same name, but with the added '_inv' at the end.

Statement 1 Setting the WeightedTAM attribute

```
context ApplicationComponent
attribute self.WeightedTAM: Real =
RegressionCoefficientPU*(use_6_inv.
    PerceivedUsefulness ->sum()/ use_6_inv.
    PerceivedUsefulness ->size())+
    RegressionCoefficientPEoU*(use_6_inv.
    PerceivedEaseOfUse->sum()/ use_6_inv.
    PerceivedEaseOfUse ->size());
```

Statement 2 Setting the Application Weight attribute

```
context UsageRelation
attribute self.ApplicationWeight: Real =
getFunctionality()/isAffected_1_inv.
    use_5_inv.Functionality;
operation getFunctionality(): Real:
self.isAffected_2.assigned_inv ->select(
    oclIsKindOf(ApplicationFunction)).
    oclAsType(ApplicationFunction).
    Functionality->sum();
```

Statement 3 Setting the WeightedTTF attribute

```
context UsageRelation
attribute self.WeightedTTF: Real =
ApplicationWeight*isAffected_1_inv.TTF*
    RegressionCoefficientTTF;
```

Statement 4 Setting the Application Component Usage attribute

```
context Application Component
attribute self.Usage: Real =
DomainConstant + WeightedTAM +
    isAffected_2_inv.WeightedTTF -> sum()
```

Statement 5 Setting the Application Component Usage attribute

```
context ProcessServiceInterface
attribute self.TTF: Real =
use_5_inv.Functionality*use_4_inv.
    TaskFulfillment
```

Statement 6 Setting the Gate_Realize Availability attribute

```
context Gate\_Realize
attribute self.Availability: Real =
if Type=true
--true means parallel
then
realize_2_inv.realize_3_inv.Availability->
    iterate( elem : Real; acc : Real = 1 |
    acc * elem )*gateToGate_realize_top.
    gateToGate_realize_bottom.Availability->
    iterate( elem : Real; acc : Real = 1 |
    acc * elem )
else
1-realize_2_inv.realize_3_inv.Availability->
    iterate( elem : Real; acc : Real = 1 |
    acc * (1-elem) )*
gateToGate_realize_top.
    gateToGate_realize_bottom.Availability->
    iterate( elem : Real; acc : Real = 1 |
    acc * (1-elem) )
endif
```

Statement 7 Setting the Gate_Use Availability attribute

```
context Gate\_Use
attribute self.Availability: Real =
if Type=true
--true means parallel
then
use_2.use_3.Availability->iterate( elem :
    Real; acc : Real = 1 | acc * elem )*
    gateToGate_use_top.gateToGate_use_bottom
    .Availability->iterate( elem : Real; acc
     : Real = 1 | acc * elem )
else
1-use_2.use_3.Availability->iterate( elem :
    Real; acc : Real = 1 | acc * (1-elem) )*
gateToGate_use_top.gateToGate_use_bottom.
    Availability->iterate( elem : Real; acc
    : Real = 1 | acc * (1-elem) )
endif
```

Statement 8 Setting the Service Availability attribute

```
context Service
attribute self.Availability: Real =
if realize_1->notEmpty()
then
realize_1.Availability
else
EvidentialAvailability
endif
```

Statement 9 Setting the Internal Behavior Element Availability attribute

```
context InternalBehaviorElement
attribute self.Availability: Real =
if use_1->notEmpty()
then
if assigned-> notEmpty()
then
use_1.Availability*assigned.Availability
else
use_1.Availability->sum()
endif
else
if assigned->notEmpty()
then
assigned.Availability
else
EvidentialAvailability
endif
endif;
```

Statement 10 Setting the Gate_Realize ResponseTime attribute

```
context Gate_Realize
attribute self.ResponseTime: Real =
if ExecutionPattern = true
--true means parallel
then
if realize_2_inv.realize_3_inv -> notEmpty()
then
--both Gate and Service
if gateToGate_realize_top -> notEmpty()
then
self.max(gateToGate_realize_top.
    WeightedResponseTime->union(
    realize_2_inv.WeightedResponseTime))
else
self.max(realize_2_inv.WeightedResponseTime)
--Only Services
endif
else
self.max(gateToGate_realize_top.
    WeightedResponseTime)
endif
else
--False Means parallel
if realize_2_inv.realize_3_inv -> notEmpty()
then
--Both Gate and Service
if gateToGate_realize_top -> notEmpty()
then
gateToGate_realize_top.WeightedResponseTime
    ->union(realize_2_inv.
    WeightedResponseTime)->sum()
else
realize_2_inv.WeightedResponseTime->sum()
--Only Services
endif
else
gateToGate_realize_top.WeightedResponseTime
    -> sum()
endif
endif;

operation max(Real): Real

input->iterate(elem : Real; acc : Real =
    input->first() | acc.max(elem) )
```

Statement 11 Setting the Realize Weighted Response time attribute

```
context Realize
attribute self.WeightedResponseTime: Real =
Weight*realize_3_inv.ResponseTime
```

Statement 12 Setting the Realize Weighted Workload attribute

```
context Realize
attribute self.WeightedWorkload: Real =
if realize_2.realize_1_inv -> notEmpty()
then
if realize_2.gateToGate_realize_bottom_inv->
    notEmpty()
then
Weight*(realize_2.realize_1_inv.Workload ->
    sum() + realize_2.
    gateToGate_realize_bottom_inv.
    WeightedWorkload ->sum())
else
Weight*(realize_2.realize_1_inv.Workload ->
    sum())
endif
else
Weight*(realize_2.
    gateToGate_realize_bottom_inv.
    WeightedWorkload ->sum())
endif
```

Statement 13 Setting the Service Response Time attribute

```
context Service
attribute self.ResponseTime: Real =
if realize_1 ->notEmpty()
then
realize_1.ResponseTime
else
EvidentialResponseTime
endif
```

Statement 14 Setting the Service Workload attribute

```
context Service
attribute self.Workload: Real =
if Use_3_inv -> notEmpty()
then
ArrivalFrequency + Use_3_inv.
    WeightedWorkload->sum()
else
ArrivalFrequency
endif
```

Statement 15 Setting the Internal Behavior Element Workload attribute

```
context InternalBehaviorElement
attribute self.Workload: Real =
if realize_3 -> notEmpty()
then
ArrivalFrequency + realize_3.
    WeightedWorkload->sum()
else
ArrivalFrequency
```

Statement 16 Setting the Internal Behavior Element Throughput attribute

```
context InternalBehaviorElement
attribute self.Throughput: Real =
ProcessingTime*(Workload)
```

Statement 17 Setting the Internal Behavior Element Processing Time attribute

```
context InternalBehaviorElement
attribute self.ProcessingTime: Real =
if use_1->notEmpty()
then
ServiceTime+ use_1.ResponseTime
else
ServiceTime
endif;
```

Statement 18 Setting the Active Structure Element Utilisation attribute

```
context ActiveStructureElement
attribute self.Utilisation: Real =
assigned_inv.Throughput-> sum())/Capacity
```

Statement 19 Setting the Internal Behavior Element Response Time

```
context InternalBehaviorElement
attribute self.ResponseTime: Real =
if assigned.Capacity = 1
then
ProcessingTime/(1-assigned.Utilisation)
else
W(Wq(Lq(P0((Workload), (1/ProcessingTime),
    assigned.Capacity), (Workload), (1/
    ProcessingTime), assigned.Capacity,
    assigned.Utilisation), (Workload)), (1/
    ProcessingTime))
endif;
                }
--below are help operations for the response
    time computations
operation P0(lambda : Real, my : Real, c :
    Real): Real =
--P0 is the probability of having zero jobs
    in the queue
1 / (P01(lambda, my, c, 0)+ (Exp((lambda/my)
    ,c)/Fact(c))*(1/(1-lambda/(c*my))))

operation Lq(p0 : Real, lambda : Real,
    my : Real, c : Real, u : Real) : Real =
--Lq is the number of requests in queue not
    being served
(p0*Exp((lambda/my),c)*u)/(Fact(c)*Exp((1-u)
    ,2))

operation Wq(lq : Real, lambda : Real) :
    Real =
-- Wq is waiting time in queue
lq/lambda;

operation W(wq : Real, my : Real) : Real =
-- W is total waiting time
wq+(1/my);

operation L(lq : Real, lambda : Real, my :
    Real) : Real =
-- L is length of queue
lq+(lambda/my);

operation Fact(x : Real) : Real =
--This is factorial
if x > 1
then
x*Fact(x-1)
else
1
endif

operation Exp(base : Real, exponent : Real)
    : Real =
-- Exp() Takes the exponent for a certain
    base (only positive values)
if exponent > 0
then
base*Exp(base, exponent-1)
else
1
endif

operation P01(lambda : Real, my : Real, c :
    Real, iterator : Int): Real =
-- P01 is a help variable
if iterator =c
then
0
else
P02(lambda, my, iterator)+P01(lambda, my, c,
    iterator+1)
endif;

operation P02(lambda : Real, my : Real, n :
    Real) : Real =
-- P02 is a help variable
(Exp((lambda/my),n))/Fact(n);
```

Statement 20 Setting the Active Structure Element Utilisation attribute

```
context PassiveComponentSet
attribute self.Accuracy: Real =
if write_Function_inv ->notEmpty()
then
precendence_inv.Accuracy*(1-
    write_Function_inv.Deterioration)+
    write_Function_inv.Correction*(1-
    precendence_inv.Accuracy)
else
if write_Service_inv ->notEmpty()
then
precendence_inv.Accuracy*(1-
    write_Service_inv.Deterioration)+
    write_Service_inv.Correction*(1-
    precendence_inv.Accuracy)
else
InputAccuracy
endif
```

Appendix B: OCL code for metamodel invariants

Statement 21 Preventing services from reading and writing to the same PassiveComponentSet

```
context Service
invariant noReadAndWrite =
not(read_Service->exists(do:
    PassiveComponentSet| write_Service->
    includes(do)))
```

Statement 22 Preventing internalbehaviorelements from reading and writing to the same PassiveComponentSet

```
context Service
invariant noWriteAndRead =
not(read_Function_inv->exists(do:
    PassiveComponentSet| write_Function->
    includes(do)))
```

Statement 23 Preventing realization relations which violate the ArchiMate constraints in terms of which services are realized by which internal behavior elements

```
context InternalBehaviorElement
invariant checkLayerMatching =

self.oclIsTypeOf(ApplicationFunction)
    implies wrapGatesAF(realize_3.realize_2,
     self.oclAsType(ApplicationFunction))
    and self.oclIsTypeOf(
    InfrastructureFunction) implies
    wrapGatesIF(realize_3.realize_2, self.
    oclAsType(InfrastructureFunction)) and
    self.oclIsTypeOf(BusinessProcess)
    implies wrapGatesBP(realize_3.realize_2,
     self.oclAsType(BusinessProcess))

--below are the operations used by
    checkLayerMatching, we only include the
    wrapGatesAF operation, wrapGatesIF and
    wrapGatesBP are very similar.

operation wrapGatesAF(gates : Gate_Realize,
    applicationFunction :
    ApplicationFunction): Boolean =
gates -> forAll(p:Gate_realize |
    checkGatesAF(p, applicationFunction));

operation checkGatesAF(gate : Gate_realize,
    internalBehaviorElement :
    ApplicationFunction):Boolean =
if gate.realize_1_inv->notEmpty() and gate.
    gateToGate_realize_bottom_inv.
    gateToGate_realize_top_inv->notEmpty()
then
checkValidCombo(gate.realize_1_inv,
    internalBehaviorElement) and gate.
    gateToGate_realize_bottom_inv.
    gateToGate_realize_top_inv->forAll(p1|
    checkGates(p1, internalBehaviorElement))
else
if gate.realize_1_inv->notEmpty()
then
checkValidCombo(gate.realize_1_inv ,
    internalBehaviorElement)
else
if gate.gateToGate_realize_bottom_inv.
    gateToGate_realize_top_inv->notEmpty()
then
gate.gateToGate_realize_bottom_inv.
    gateToGate_realize_top_inv->forAll(p1|
    checkGates(p1, internalBehaviorElement))
else
true
endif
endif
endif

operation checkValidCombo(services : Service
    , internalbehaviorelement :
    InternalBehaviorElement): Boolean =
if internalbehaviorelement.oclIsKindOf(
    ApplicationFunction)
then
services ->forAll(p1 |
p1.oclIsKindOf(ApplicationService))
else
if internalbehaviorelement.oclIsTypeOf(
    InfrastructureFunction)
then
services ->forAll(p1 |
p1.oclIsKindOf(InfrastructureService))
else
if internalbehaviorelement.oclIsKindOf(
    BusinessProcess)
then
services ->forAll(p1 |
p1.oclIsKindOf(BusinessService))
else
true
endif
endif
endif
```

Appendix C: Instructions for using the EAAT tool

In order to comprehend the models that are presented in the submission you need to use the enterprise architecture analysis tool. The following describes the usage with a Windows operating system:

1. Download the file "SoSyM-GUPM.zip" from http://www.ics.kth.se/eat/SoSyMGUPM.zip.
2. Extract the zip file "SoSyMGUPM.zip".
3. In the created folder you find two subfolders (CM containing the tool and Model containing the presented model).
4. If you are using Windows with limited user privileges, please copy the jsmile.dll to be found in the CM folder into your C:\windows\system32 folder (you must have administrator privileges to do so).
5. Execute the runCM.bat file that is included in the CM folder.
6. In the "Load Abstract Model File/ previously created Model" dialog that is shown on start click on the browse button and navigate to the Model folder, which was included in the zip file. Select "EntireGUPMShowcase.iEaat" and press the ok button.
7. Now the Enterprise Architecture Analysis Tool is shown allowing to consider both models and meta model used within the paper.
8. To the left you find "Views" that correspond to the viewpoints presented in Sect. 5 of the submission.
9. Below you find the "Meta model" and the "Viewpoints" as they have been described in Sects. 4 and 5.
10. Right-clicking on entities of the models shows details and allows to trace how attribute values are calculated.
11. The models can be calculated by clicking the "Calculate" button at the upper right-side of the tool.

References

1. Object constraint language, version 2.2. Tech. rep., Object Management Group, OMG (2010). http://www.omg.org/spec/OCL/2.2. OMG Document Number: formal/2010-02-01
2. Van der Aalst, W., Van Hee, K.: Business process redesign: a Petri-net-based approach. Comput. Ind. **29**(1–2), 15–26 (1996)
3. Aier, S., Fischer, C.: Criteria of progress for information systems design theories. Inf. Syst. E Bus. Manag. **9**(1), 133–172 (2011)
4. Ballou, D.P., Pazer, H.L.: Modeling data and process quality in multi-input, multi-output information systems. Manag. Sci. **31**(2), 150–162 (1985). http://www.jstor.org/stable/2631512
5. Barber, K., Graser, T., Holt, J.: Enabling iterative software architecture derivation using early non-functional property evaluation. In: Proceedings of the 17th IEEE International conference on Automated Software Engineering (ASE), pp. 172–182 (2002)

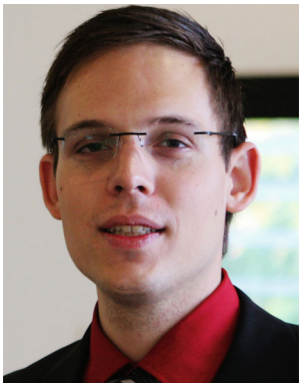
6. Batini, C., Scannapieco, M.: Data Quality: Concepts, Methodologies and Techniques. Springer, New York (2006)
7. Bengtsson, P., Bosch, J.: Scenario-based software architecture reengineering. In: Proceedings of the Fifth International Conference on Software Reuse, 1998, pp. 308–317. IEEE (2002)
8. Bharadwaj, A., Keil, M., Mähring, M.: Effects of information technology failures on the market value of firms. J. Strateg. Inf. Syst. **18**(2), 66–79 (2009)
9. Bocciarelli, P., D'Ambrogio, A.: A model-driven method for describing and predicting the reliability of composite services. Softw. Syst. Model. **10**, 265–280 (2011). doi:10.1007/s10270-010-0150-3
10. Bradley, R.V., Pratt, R.M.E., Byrd, T.A., Simmons, L.: The role of enterprise architecture in the quest for it value. MIS Q. Exec. **10**(2), 73–80 (2011)
11. Brynjolfsson, E., Hitt, L.: Paradox lost? firm-level evidence on the returns to information systems spending. Manag. Sci. **42**(4), pp. 541–558 (1996). http://www.jstor.org/stable/2634387
12. Buchholz, P.: A class of hierarchical queueing networks and their analysis. Queueing Syst. **15**(1), 59–80 (1994)
13. Buckl, S., Ernst, A., Matthes, F., Schweda, C.: An information model for landscape management-discussing temporality aspects. In: Proceedings of the Service-Oriented Computing-ICSOC 2008 Workshops pp. 363–374 (2009)
14. Buckl, S., Franke, U., Holschke, O., Matthes, F., Schweda, C., Sommestad, T., Ullberg, J.: A pattern-based approach to quantitative enterprise architecture analysis. In: 15th Americas Conference on Information Systems (AMCIS). Association for, Information Systems (2009)
15. Burton, B., Allega, P.: Hype Cycle for Enterprise Architecture, 2010. GARTNER, vol. G00201646 (2010)
16. Buschle, M., Ullberg, J., Franke, U., Lagerström, R., Sommestad, T.: A tool for enterprise architecture analysis using the prm formalism. Information Systems Evolution. Lecture Notes in Business Information Processing, vol. 72, pp. 108–121 (2011)
17. Buzen, J.: Computational algorithms for closed queueing networks with exponential servers. Commun. ACM **16**(9), 527–531 (1973)
18. Buzen, J.: Fundamental operational laws of computer system performance. Acta Informatica **7**(2), 167–182 (1976)
19. Chang, H.: Intelligent agent's technology characteristics applied to online auctions' task: A combined model of TTF and TAM. Technovation **28**(9), 564–577 (2008)
20. Cobb, B.R., Shenoy, P.P.: Inference in hybrid bayesian networks with mixtures of truncated exponentials. Int. J. Approx. Reason. **41**(3), 257–286 (2006). doi:10.1016/j.ijar.2005.06.002
21. Cortellessa, V., Singh, H., Cukic, B.: Early reliability assessment of uml based software models. In: WOSP '02: Proceedings of the 3rd international workshop on Software and performance, pp. 302–309. ACM, New York, NY, USA (2002). doi:10.1145/584369.584415
22. Cushing, B.E.: A mathematical approach to the analysis and design of internal control systems. Acc. Rev. **49**(1), 24–41 (1974). http://www.jstor.org/stable/244795
23. Davis, F.D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q. **13**(3), 319–340 (1989). http://www.jstor.org/stable/249008
24. De Boer, F., Bonsangue, M., Jacob, J., Stam, A., Van der Torre, L.: Enterprise architecture analysis with xml. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences 2005, HICSS'05, p. 222b. IEEE (2005)
25. DeLone, W., McLean, E.: Information systems success: the quest for the dependent variable. Inf. Syst. Res. **3**(1), 60–95 (1992)
26. DeLone, W., McLean, E.: The DeLone and McLean model of information systems success: A ten-year update. J. Manag. Inf. Syst. **19**(4), 9–30 (2003)
27. Demathieu, S., Thomas, F., André, C., Gérard, S., Terrier, F.: First experiments using the uml profile for marte. In: 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), 2008, pp. 50–57. IEEE (2008)
28. Dishaw, M., Strong, D.: Supporting software maintenance with software engineering tools: A computed task-technology fit analysis. J. Syst. Softw. **44**(2), 107–120 (1998)
29. Dishaw, M., Strong, D.: Extending the technology acceptance model with task-technology fit constructs. Inf. Manag. **36**(1), 9–21 (1999)
30. DoD: Dod architecture framework version 2.0. United States Department of Defense (2009)
31. Dunsire, K., O'Neill, T., Denford, M., Leaney, J.: The ABACUS architectural approach to computer-based system and enterprise evolution. In: Proceedings of the 12th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, p. 69. IEEE Computer Society (2005)
32. Ernst, A.: Enterprise architecture management patterns. In: Proceedings of the 15th Conference on Pattern Languages of Programs, p. 7. ACM (2008)
33. Framework, E.M.: EMF: OCL plugin for the Eclipse Modeling Framework (2011). http://www.eclipse.org/emf/
34. Frank, U., Heise, D., Kattenstroth, H., Schauer, H.: Designing and utilising business indicator systems within enterprise models-outline of a method. Modellierung betrieblicher Informationssysteme (MobIS, 2008) : November, Saarbrücken. Germany, pp. 89–105 (2008)
35. Franke, U., Flores, W.R., Johnson, P.: Enterprise architecture dependency analysis using fault trees and bayesian networks. 42nd Annual Simulation Symposium (ANSS) (2009)
36. Franke, U., Johnson, P., Ericsson, E., Flores, W.R., Zhu, K.: Enterprise architecture analysis using fault trees and MODAF. In: Proceedings of the CAiSE Forum 2009, vol. 453, pp. 61–66 (2009). ISSN: 1613–0073
37. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: International Joint Conference on Artificial Intelligence, vol. 16, pp. 1300–1309. Citeseer (1999)
38. Gammelgård, M., Ekstedt, M., Närman, P.: A method for assessing the business value of information system scenarios with an estimated credibility of the result. Int. J. Serv. Technol. Manag. **13**(1), 105–133 (2010)
39. Gilmore, S., Gönczy, L., Koch, N., Mayer, P., Tribastone, M., Varró, D.: Non-functional properties in the model-driven development of service-oriented systems. Softw. Syst. Model. **10**, 287–311 (2011). doi:10.1007/s10270-010-0155-y
40. Glazner, C.: Understanding enterprise behavior using hybrid simulation of enterprise architecture. Ph.D. thesis, Massachusetts Institute of Technology (2009)
41. Goodhue, D., Thompson, R.: Task-technology fit and individual performance. Mis Q. **19**(2), 213–236 (1995)
42. Gregor, S., Jones, D.: The anatomy of a design theory. J. Assoc. Inf. Syst. **8**(5), 312–335 (2007)
43. Harrison, P., Patel, N.: Performance Modelling of Communication Networks and Computer Architectures. International Computer Science. Addison-Wesley Longman Publishing Co., Inc., Boston (1992)
44. Henderson, J., Cooprider, J.: Dimensions of I/S planning and design aids: a functional model of CASE technology. Inf. Syst. Res. **1**(3), 227 (1990)
45. Hevner, A., March, S., Park, J., Ram, S.: Design science in information systems research. Mis Q. **28**(1), 75–105 (2004)
46. Hillier, F., Lieberman, G.: Introduction to Operations Research Eighth. McGraw-Hill, Singapore (2005)
47. Holschke, O., Närman, P., Flores, W., Eriksson, E., Schönherr, M.: Using enterprise architecture models and bayesian belief networks

for failure impact analysis. In: Service-Oriented Computing-ICSOC 2008 Workshops, pp. 339–350. Springer (2009)
48. Høyland, A., Rausand, M.: System Reliability Theory: Models and Statistical Methods. Wiley, New York (1994)
49. Iacob, M., Jonkers, H.: Quantitative analysis of enterprise architectures. Interoperability of Enterprise Software and Applications pp. 239–252 (2006)
50. Iacob, M., Jonkers, H.: Quantitative analysis of service-oriented architectures. Int. J. Enterp. Inf. Syst. **3**(1), 42–60 (2007)
51. Iacob, M.E., Jonkers, H.: Analysis of enterprise architectures. Tech. Rep, Telematica Instituut (TI) (2004)
52. IBM Global Services: Improving systems availability. Tech. Rep, IBM Global Services (1998)
53. IEC technical committee 57: Iec 61968-1—application integration at electric utilities—system interfaces for distribution management—part 1: Interface architecture and general requirements (2003)
54. ISO: Iso/iec 9126–2:2003 software engineering—product quality—part 2: External metrics (2003)
55. IT Governance Institute: Cobit 4.1. ISACA (2007). http://www.isaca.org/Knowledge-Center/cobit/Pages/Downloads.aspx
56. Jaeger, M.C., Rojec-Goldmann, G., Muhl, G.: Qos aggregation in web service compositions. In: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service, 2005. EEE'05, pp.181–185 (2005)
57. Jain, R.: The Art of Computer Systems Performance Analysis : Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley, New York (1991)
58. Johansson, E., Ekstedt, M., Johnson, P.: Assessment of enterprise information security: the importance of information search cost. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences, 2006. HICSS'06, pp. 219a–219a (2006)
59. Johansson, K.: Driftsäkerhet och underhåll. Studentlitteratur, Lund (1997)
60. Johnson, P., Ekstedt, M.: Enterprise Architecture: Models and Analyses for Information Systems Decision Making. Studentlitteratur, Lund (2007)
61. Johnson, P., Lagerström, R., Närman, P., Simonsson, M.: Enterprise architecture analysis with extended influence diagrams. Inf. Syst. Front. **9**(2), 163–180 (2007)
62. Kans, M., Ingwald, A.: Analysing it functionality gaps for maintenance management. In: Engineering Asset Lifecycle Management. Proceedings of the 4th World Congress of Engineering Asset Management (WCEAM) 2009. The original pulication is available at http://www.springerlink.com
63. Kasunic, M.: Measuring systems interoperability: challenges and opportunities. Tech. Rep, DTIC Document (2001)
64. Kazman, R., Bass, L., Webb, M., Abowd, G.: SAAM: A method for analyzing the properties of software architectures. In: Proceedings of the 16th International Conference on Software Engineering, pp. 81–90 (1994)
65. Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., Carriere, J.: The architecture tradeoff analysis method. In: Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems, 1998. ICECCS'98, pp. 68–78 (2002)
66. Klein, M., Kazman, R., Bass, L., Carriere, J., Barbacci, M., Lipson, H.: Attribute-based architecture styles. In: Software Architecture: TC2 First Working IFIP Conference on Software Architecture (WICSA1): 22–24 February 1999, San Antonio, Texas, USA, p. 225. Kluwer Academic Publishing, Dordrecht (1999)
67. Klopping, I., McKinney, E.: Extending the technology acceptance model and the task and the task-technology fit model to technology fit model to consumer E Consumer E-Commerce. Inf. Technol. Learn. Perform. J. **22**(1), 35 (2004)
68. Kurpjuweit, S., Winter, R.: Viewpoint-based meta model engineering. EMISA **2007**, 143 (2007)
69. Lagerström, R., Johnson, P., Höök, D.: Architecture analysis of enterprise systems modifiability-models, analysis, and validation. J. Syst. Softw. **83**(8), 1387–1403 (2010)
70. Lam, S., Lien, Y.: A tree convolution algorithm for the solution of queueing networks. Commun. ACM **26**(3), 215 (1983)
71. Lankhorst, M.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, Berlin (2009)
72. Lankhorst, M., Proper, H., Jonkers, H.: The Architecture of the ArchiMate Language. In: Enterprise, Business-Process and Information Systems Modeling: 10th International Workshop, Bpmds 2009, and 14th International Conference, Emmsad 2009, Held at Caise 2009, Amsterdam, the Netherlands, June 8–9, 2009, Proceedings, p. 367. Springer (2009)
73. Laprie, J., Kanoun, K.: X-ware reliability and availability modeling. IEEE Trans. Softw. Eng. **18**(2), 130–147 (2002)
74. Leangsuksun, C., Shen, L., Liu, T., Song, H., Scott, S.: Availability prediction and modeling of high mobility OSCAR cluster. In: Proceedings of the 2003 IEEE International Conference on Cluster Computing, 2003, pp. 380–386. IEEE (2005)
75. Ledoux, J.: Availability modeling of modular software. IEEE Trans. Reliab. **48**(2), 159–168 (2002)
76. Lee, Y.W., Pipino, L.L., Funk, J.D., Wang, R.Y.: Journey to Data Quality. MIT Press, Cambridge (2006)
77. Lankhorst, M.M.: Enterprise architecture modelling-the issue of integration. Adv. Eng. Inform. **18**(4), 205–216 (2004). doi:10.1016/j.aei.2005.01.005
78. March, S.T., Smith, G.F.: Design and natural science research on information technology. Decis. Support Syst. **15**(4), 251–266 (1995). doi:10.1016/0167-9236(94)00041-2
79. Melville, N., Kraemer, K., Gurbaxani, V.: Review: information technology and organizational performance: An integrative model of it business value. MIS Q. **28**(2), pp. 283–322 (2004). http://www.jstor.org/stable/25148636
80. de Miguel, M., Lambolais, T., Hannouz, M., Betgé-Brezetz, S., Piekarec, S.: UML extensions for the specification and evaluation of latency constraints in architectural models. In: Proceedings of the 2nd international workshop on Software and performance, pp. 83–88. ACM (2000)
81. Ministry of Defence: MODAF Handbook, technical specification for MODAF. Ministry of Defence (2005)
82. Montgomery, D.: Design and Analysis of Experiments. Wiley, New York (1991)
83. Mukhopadhyay, T., Kekre, S., Kalathur, S.: Business value of information technology: A study of electronic data interchange. MIS Q. **19**(2), pp. 137–156 (1995). http://www.jstor.org/stable/249685
84. Närman, P., Buschle, M., König, J., Johnson, P.: Hybrid probabilistic relational models for system quality analysis. In: Enterprise Distributed Object Computing Conference 2010, EDOC '10: Vitoria, ES, Brazil. 14th International IEEE. IEEE (2010) (inpress)
85. Närman, P., Holm, H., Ekstedt, M., Honeth, N.: An interview-based enterprise architecture analysis method for service response time predictions. J. Strateg. Inf. Syst. (2011) (submitted)
86. Närman, P., Holm, H., Höök, D., Honeth, N., Johnson, P.: Using enterprise architecture and technology adoption models to predict application usage. J. Syst. Softw. **85**(8), 1953–1967 (2012)
87. Närman, P., Holm, H., Johnson, P., König, J., Chenine, M., Ekstedt, M.: Data accuracy assessment using enterprise architecture. Enterp. Inf. Syst. **5**(1), 37–58 (2011)
88. Närman, P., Johnson, P., Lagerström, R., Franke, U., Ekstedt, M.: Data collection prioritization for system quality analysis. Electron. Notes Theor. Comput. Sci. **233**, 29–42 (2009)

89. NATO Consultation, Command and Control Board: Command and control board (nc3b):"Snato c3 system architecture framework. Tech. Rep., AC/322-WP/0125, Brussels (2007)
90. Närman, P., Franke, U., König, J., Buschle, M., Ekstedt, M.: Enterprise architecture availability analysis using fault trees and stakeholder interviews. Enterp. Inf. Syst. **0**(0), 1–25 (2012). doi:10.1080/17517575.2011.647092. Available online 31 Jan 2012
91. Nunamaker, J., Chen, M., Purdin, T.: Systems development in information systems research. J. Manag. Inf. Syst. **7**(3), 89–106 (1990)
92. Pagani, M.: Determinants of adoption of high speed data services in the business market: Evidence for a combined technology acceptance model with task technology fit model. Inf. Manag. **43**(7), 847–860 (2006). doi:10.1016/j.im.2006.08.003
93. Palmer, J.: Web site usability, design, and performance metrics. Inf. Syst. Res. **13**(2), 151–167 (2003)
94. Patton, J.D.: Preventive maintenance. Instrument Society of America, New York (1983)
95. Bernus, P.: Enterprise models for enterprise architecture and iso9000:2000. Annu. Rev. Control **27**(2), 211–220 (2003). doi:10.1016/j.arcontrol.2003.09.004
96. Petriu, D., Wang, X.: From UML descriptions of high-level software architectures to LQN performance models. In: Applications of Graph Transformations with Industrial Relevance, Lecture Notes in Computer Science, vol. 1779/2000, pp. 217–221 (2000). doi:10.1007/3-540-45104-8_4
97. Raderius, J., Närman, P., Ekstedt, M.: Assessing system availability using an enterprise architecture analysis approach. In: Proceedings of 3rd Workshop on Trends in Enterprise Architecture Research (TEAR 2008), Sydney, Australia. Springer (2009)
98. Redman, T.C.: Data Quality for the Information Age. Artech House, Boston (1996)
99. Redman, T.C.: The impact of poor data quality on the typical enterprise. Commun. ACM **41**, 79–82 (1998). doi:10.1145/269012.269025
100. Reiser, M., Lavenberg, S.: Mean-value analysis of closed multichain queuing networks. J. ACM (JACM) **27**(2), 313–322 (1980)
101. Reussner, R., Schmidt, H., Poernomo, I.: Reliability prediction for component-based software architectures. J. Syst. Softw. **66**(3), 241–252 (2003)
102. Riempp, G., Gieffers-Ankel, S.: Application portfolio management: a decision-oriented view of enterprise architecture. Inf. Syst. E Bus. Manag. **5**(4), 359–378 (2007)
103. Rodrigues, G., Rosenblum, D., Uchitel, S.: Using scenarios to predict the reliability of concurrent component-based software systems. Lecture Notes in Computer Science, vol. 3442, pp. 111–126 (2005). doi:10.1007/978-3-540-31984-9_9
104. Ross, J., Beath, C.: Sustainable it outsourcing success: Let enterprise architecture be your guide. MIS Q. Exec. **5**(4), 8–92 (2006)
105. Ross, J., Weill, P., Robertson, D.: Enterprise Architecture as Strategy: Creating a Foundation for Business Execution. Harvard Business Press, Boston (2006)
106. Sasa, A., Krisper, M.: Enterprise architecture patterns for business process support analysis. J. Syst. Softw. **84**(9): 1480–1506 (2011)
107. Scott, D.: How to Assess Your IT Service Availability Levels (2009)
108. Simon, D., Fischbach, K., Schoder, D.: Application portfolio management-an integrated framework and a software tool evaluation approach. Comm. Assoc. Inf. Syst. **26**(1), 3 (2010)
109. Smit, K., Slaterus, W.: Information Model for Maintenance Management (IMMM). Cap Gemini Publishing, Rijswijk (1922)
110. Smith, C., Williams, L.: Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Addison-Wesley, Boston (2002)
111. Sommestad, T., Ekstedt, M., Johnson, P.: A probabilistic relational model for security risk analysis. Comput. Secur (2010)
112. Stamatelatos, M., Vesely, W., Dugan, J., Fragola, J., Minarick, J., Railsback, J.: Fault tree handbook with aerospace applications (2002). http://www.hq.nasa.gov/office/codeq/doctree/fthb.pdf
113. International Organization for Standardization.: Iso/iec 42010:2007—systems and software engineering—recommended practice for architectural description of software-intensive systems. JTC 1/SC 7—Softw. Syst. Eng. (2007)
114. The Open Group: Archimate 1.0 specification. Van Haren Publishing, Zaltbommel (2009)
115. The Open Group: Togaf version 9 "enterprise edition" (2009)
116. Ullberg, J., Franke, U., Buschle, M., Johnson, P.: A tool for interoperability analysis of enterprise architecture models using pi-OCL. In: Proceedings of the International Conference on Interoperability for Enterprise Software and Applications (I-ESA) (2010)
117. Ullberg, J., Lagerström, R., Johnson, P.: A framework for service interoperability analysis using enterprise architecture models. In: IEEE International Conference on Services Computing (2008)
118. Van Bon, J.: Foundations of IT Service Management based on ITIL. Van Haren Publishing, Zaltbommel (2007)
119. Venkatesh, V., Bala, H., Venkatraman, S., Bates, J.: Enterprise architecture maturity: The story of the veterans health administration. MIS Q. Exec. **6**(2), 79–90 (2007)
120. Venkatesh, V., Morris, M.G., Davis, G.B., Davis, F.D.: User acceptance of information technology: Toward a unified view. MIS Q. **27**(3), pp. 425–478 (2003). http://www.jstor.org/stable/30036540
121. Venkatraman, N.: The concept of fit in strategy research: Toward verbal and statistical correspondence. Acad. Manag. Rev. **14**(3), 423–444 (1989)
122. Vessey, I.: Expertise in debugging computer programs: An analysis of the content of verbal protocols. Systems, Man and Cybernetics, IEEE Transactions on **16**(5), 621–637 (1986). doi:10.1109/TSMC.1986.289308
123. Walls, J., Widmeyer, G., El Sawy, O.: Building an information system design theory for vigilant eis. Inf. Syst. Res. **3**(1), 36–59 (1992)
124. Wang, R., Ziad, M., Lee, Y.: Data Quality. Kluwer Academic Publishing, Dordrecht (2001)
125. Wang, R.Y.: Information Quality. M. E. Sharpe, Armonk (2005)
126. Wang, W., Wu, Y., Chen, M.: An architecture-based software reliability model. In: Dependable Computing, 1999. Proceedings. 1999 Pacific Rim International Symposium on, pp. 143–150. IEEE (2002)
127. Wang, W.L., Pan, D., Chen, M.H.: Architecture-based software reliability modeling. J. Syst. Softw. **79**(1), 132–146 (2006). doi:10.1016/j.jss.2005.09.004
128. Weill, P., Vitale, M.: Assessing the health of an information systems applications portfolio: An example from process manufacturing. MIS Q. **23**(4), 601–624 (1999)
129. Woodhouse, J.: Putting the total jigsaw puzzle together: PAS 55 standard for the integrated, optimized management of assets. In: International Maintenance Conference (2006)
130. Yacoub, S., Cukic, B., Ammar, H.: A scenario-based reliability analysis approach for component-based software. Reliability, IEEE Transactions on **53**(4), 465–480 (2004). doi:10.1109/TR.2004.838034
131. Zachman, J.: A framework for information systems architecture. IBM Syst. J. **38**(2/3), 454–470 (1999)

Author Biographies

Per Närman is a PhD student at the department of Industrial Information and Control Systems at the Royal Institute of Technology (KTH) in Stockholm, Sweden. He has published several journal and conference papers on the topic of enterprise architecture analysis and IT management. As of 2011 he is also employed as a management consultant at Capgemini Consulting.



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

Mathias Ekstedt is Associate Professor at the Royal Institute of Technology (KTH) in Stockholm, Sweden. His research interests include systems and enterprise architecture modelling and analysis in particular with respect to information and cyber security.