Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 233 (2009) 29–42
www.elsevier.com/locate/entcs

Data Collection Prioritization for System Quality Analysis

Per Närman [1] Pontus Johnson[2] Robert Lagerström[3]
Ulrik Franke[4] Mathias Ekstedt[5]

Industrial Information and Control Systems
Royal Institute of Technology
Stockholm, Sweden

Abstract

When assessing software quality the cost of collecting the data needed for analysis is often quite substantial. This paper proposes the use of Bayesian networks for assessing software qualities combined with an algorithm for how to prioritize which data to collect in order to minimize the cost of the assessment. This algorithm, the Diagnosis algorithm, is implemented in the Bayesian network tool 'GeNIe'. An example evaluation of service interoperability in the paper demonstrates that using the algorithm may reduce time spent on data collection significantly.

Keywords: System Quality Analysis, Bayesian Networks, Data collection Strategies

1 Introduction

In recent years, software, system and enterprise architecture have become established disciplines in both industry and academia. Architecture models may aid the communication between various stakeholders. Architecture models may also aid the understanding of the complex systems they represent. One important part of understanding is to be able to infer new knowledge from a model, i.e. to be able to analyze the models. As an example, by considering an architecture model over a set of enterprise services and their relations, an observer may infer that these services will be unable to interoperate (perhaps due to a protocol mismatch). The model does not explicitly state that the services cannot interoperate, but by using a set

[1] Email: pern@ics.kth.se
[2] Email: pj101@ics.kth.se
[3] Email: robertl@ics.kth.se
[4] Email: ulrikf@ics.kth.se
[5] Email: mathiase@ics.kth.se

doi:10.1016/j.entcs.2009.02.059

of inference rules (e.g. that the protocols must match), it is possible to deduce this new information.

This paper builds on previous research within the field of architecture analysis where architecture models are analyzed using a formalism based on Bayesian statistics [9] [10]. This approach allows the analysis of various system properties, such as the interoperability, information security and the availability of software systems. In this paper we will use enterprise service interoperability analysis as a running example.

Modeling of large systems, however, may become prohibitively expensive, as models grow very large. Therefore, it is desirable to be able to perform analyses on incomplete models, i.e. on models that do not contain all the information required for a perfectly credible assessment of the considered property (in our example interoperability). Instead, it is desirable to obtain not only analysis results, but also estimates of their uncertainty or lack of credibility. Using the proposed Bayesian statistics-based approach, it is possible to obtain results stating that the probability of this system being able to interoperate is, for instance, 80%. In order to improve the credibility of the results, the architecture models need to be refined.

The refinement of architecture models generally requires data collection. Someone needs to find and study documentation and code and perhaps interview developers and other stakeholders. Since this data collection is costly, it is desirable to collect the most important data, or evidence, first. In this paper, we propose a method for determining the order in which various pieces of evidence should be collected.

The next section provides some background to the architecture analysis approach on which this paper is based. Section 3 concretizes the approach in the case of enterprise service interoperability analysis. In the fourth section, an example system is modeled. It is demonstrated that the model rapidly becomes very large and that there is a need for a data collection strategy. In Section 5, an algorithm for selecting the most preferable next piece of evidence is proposed. Because the algorithm requires information on the cost of evidence collection, Section 6 considers how such costs may be assigned. In Section 7, the results of using the algorithm are demonstrated. Section 8 concludes the paper.

2 Abstract and concrete models

An abstract model is an enterprise architecture metamodel containing entities and entity relations, augmented with the attributes and attribute relations of a Bayesian network.

Entities are fundamental parts found in most metamodels. Entities represent the objects of interest when modeling, e.g. systems, services, persons, or processes. Entities in abstract models are similar to classes found in UML [15].

Entity relations connect two entities, e.g. "Service Description describes Service" or "Person is a resource of a Process". Entity relations also state the multiplicity of the relationship between the entities, e.g. that one person can be the resource of

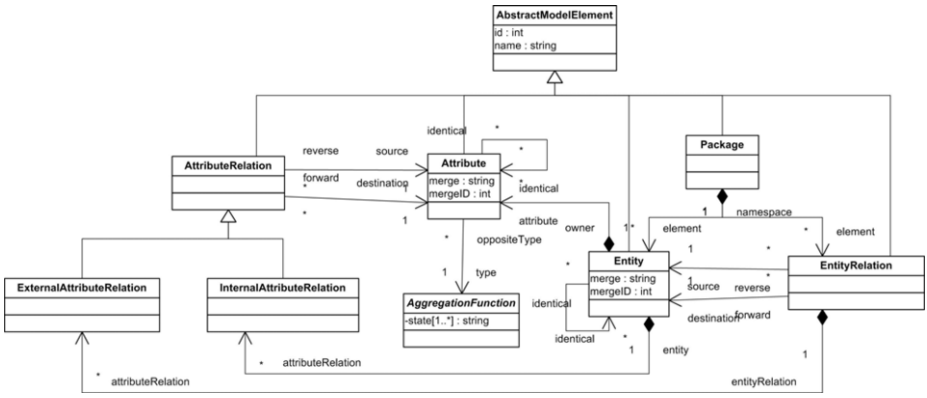

Fig. 1. A UML description of an abstract model.

zero or more processes.

Attributes of an abstract model represent random variables related to the entities. UML also have attributes related to entities, but the attributes in abstract models differ from the attributes in UML. In abstract models, attributes and attribute relations represent the nodes and relations of a Bayesian network, see below. A UML description of abstract models is given in Figure 1. A richer account of abstract and concrete models is found in Johnson et al. [11].

2.1 Bayesian networks

Friedman et al. [5] describes a Bayesian network, $B = (G, P)$, as a representation of a joint probability distribution, where $G = (V, E)$ is a directed acyclic graph consisting of vertices, V, and edges, E. The vertices denote a domain of random variables $X_1, \ldots, X_n$, also called chance nodes. In the context of abstract models, each chance node corresponds to an attribute. Each chance node, X_i, may assume a value x_i from the finite domain $Val(X_i)$. The edges denote causal dependencies between the nodes, i.e. the causal relations between the nodes. The second component, P, of the network B, describes a conditional probability distribution for each chance node, $P(X_i)$, given its parents $\mathrm{Pa}(X_i)$ in G. It is possible to write the joint probability distribution of the domain $X_1, \ldots, X_n$ using the chain rule of probability, in the product form.

$$P\left(X_1, ..., X_n\right) = \prod_{i=1}^{n} P\left(X_i | \mathrm{Pa}\left(X_i\right)\right).$$

In order to specify the joint distribution, the respective conditional probabilities that appear in the product form must be defined. The second component P describes distributions for each possible value x_i of X_i, and $\mathrm{pa}(X_i)$ of $\mathrm{Pa}(X_i)$, where $\mathrm{pa}(X_i)$ is the set of values of $\mathrm{Pa}(x_i)$. These conditional probabilities are represented in matrices, here forth called Conditional Probability Matrices (CPMs). Using a Bayesian network, it is possible to answer questions such as what is the probability of variable X being in state x_1 given that $Y = y_2$ and $Z = z_1$.

	Language A		Language B	
	Language A	Language B	Language A	Language B
Compatible	1	0	0	1
Incompatible	0	1	1	0

Table 1
A conditional probability matrix for the Equality relation.

In the general case, the relations between variables described by the conditional probability matrices can be arbitrarily complicated conditional probabilities. The example model used in this paper uses only a few rather simple relations, viz. AND, Weighted sampling distribution, and Equality. An example CPM for the Equality relation is given in Table 1.

More comprehensive treatments on Bayesian networks can be found in e.g. Neapolitan [14], Jensen [7], Shachter [20] and Pearl [18].

2.2 Creating concrete models

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

An abstract model can be iteratively improved using the well known learning algorithms for Bayesian networks on evidence collected in case studies [7]. The result of this learning process determines the prior values of all attributes, and the conditional probability matrices in the final abstract model.

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

3 An abstract model for interoperability analysis

In this section, we present an abstract model suggested for enterprise service interoperability analysis. This abstract model is a simplified version of a model presented in Ullberg et al [21]. The abstract model contains seven entities all briefly described together with the accompanying attributes, see Figure 2.

`Services` are independent building blocks that collectively represent an application environment, much like components of a software system. However, services possess a number of qualities that components lack, e.g. the complete autonomy from other services. This allows a service to be responsible for its own domain. Furthermore, services are typically limited in their scope to support a specific business function or a group of related functions [3] Each service at an enterprise need to be of high quality, i.e. every service needs the foundation to be able to interoperate with other services. Two aspects affecting service quality are the correctness and availability of a service.

Services have `service descriptions`. These are used for advertising and describing the service capabilities, behavior, quality, and its interface [16]. The service descriptions have four attributes: understandability, completeness, correctness, and existence in service repository. These attributes all affect the quality of the service being described.

For communication among services, each service has a `service interface`. The service interface contains the protocols a service needs for communication and it specifies which operations the service provides or invokes [4][17].

Services use a `service bus`, often referred to as an enterprise service bus (ESB), as a communication medium. The service bus is a middleware-like solution to manage message and transaction traffic [13]. `The service orchestration description` is the specification that details and controls the orchestration of services to interact [17] These descriptions are written in a `service orchestration language`, where BPEL (Business Process Execution Language) is considered an industry standard. The services orchestration description must be service compatible in order for the orchestrated services to be of high quality. Services interoperate in `service clusters`, e.g. three fine-grained services A1, A2, and A3 may be orchestrated to provide a more coarse-grained service B. Clusters thus appear on many different levels of abstraction, with the most coarse-grained enterprise service consisting of several other, more fine-grained, clusters.

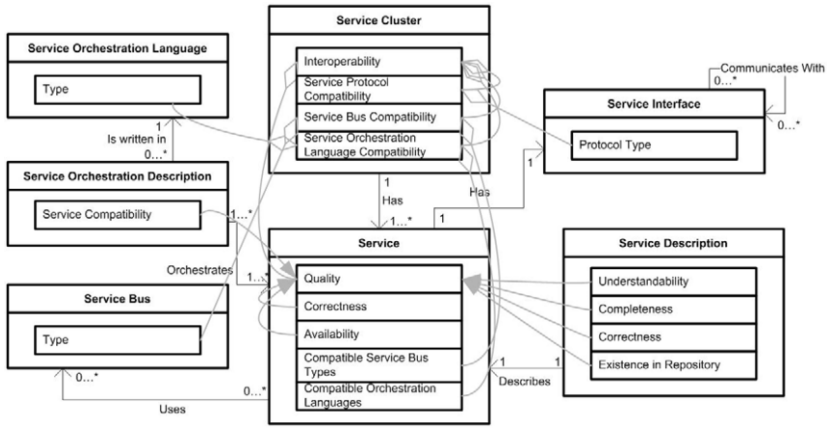

Fig. 2. The abstract model for service cluster interoperability.

All clusters, independent of abstraction level, can be analyzed with respect to interoperability; i.e. how well the services within the cluster interoperate. The service cluster interoperability is measured in terms of the quality of each service within the cluster, the service protocol compatibility, service bus compatibility, and service orchestration language compatibility

4 Example concrete model

Using the abstract model for service cluster interoperability we now proceed to create an example concrete model of a very small `service cluster` consisting of

merely two `services`. The purpose of this concrete model is to assess the interoperability of the `service cluster`.

4.1 Inflating the concrete model

The creation of the concrete model commences by setting the multiplicities indicated in the abstract model, followed by the naming of the instantiated entities in the concrete model. Here, we have one `service cluster` with two `services`: 'Work order initiation' and 'Work order closing'. Each `service` has one `service description` and one `service interface` which are communicating with each other. The integration of the services is managed by one `service bus` and one `service orchestration description`. An overview of the concrete model can be found in Figure 3.

Each object of the concrete model has a number of attributes; questions that need to be answered for the interoperability assessment. There are several ways for us to collect evidence for each attribute; we can therefore find several pieces of evidence about a single attribute. Viable ways to collect information are, for instance, to interview people, perform manual tests, study documents, or a combination of these. In the current example, we introduce three sources of evidence. Max the consultant, who is in charge of developing the service cluster. George, the system owner, is responsible for the system that implements the services. Finally, the investigator can collect information by performing manual inspections of the various attributes in the concrete model.

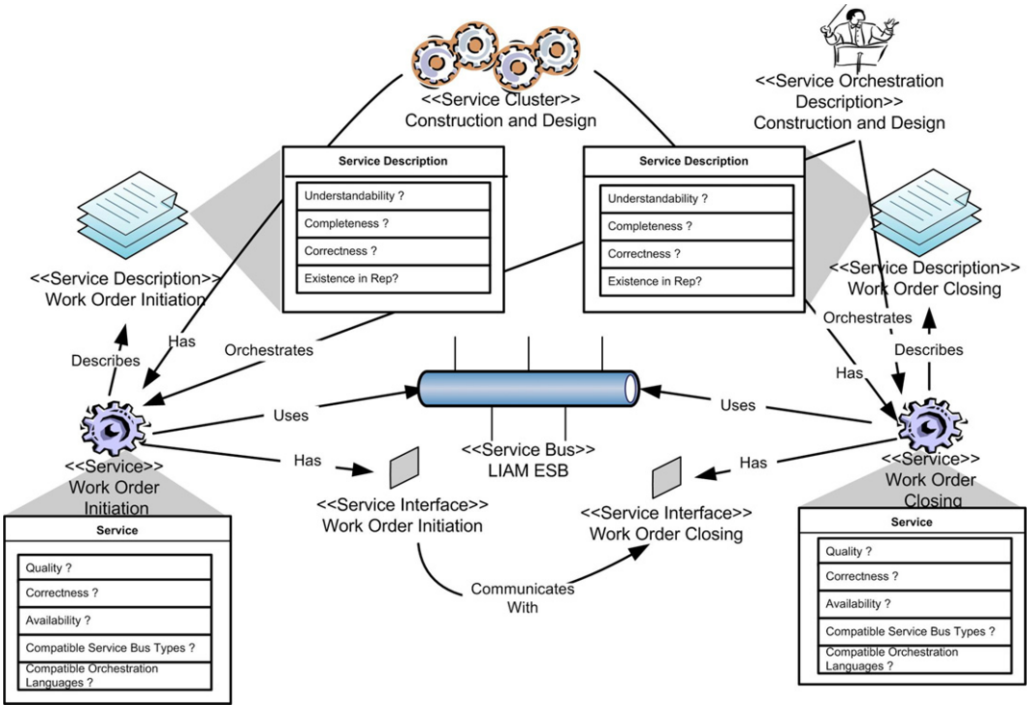

Fig. 3. A concrete model of a service cluster of two services with some sample attributes.

Max's answer		Service Description Completeness	
		Complete	Incomplete
	Complete	0.9	0.1
	Incomplete	0.1	0.9

Table 2
The credibility of a piece of evidence specified in a conditional probability matrix. The probability of Max's answer being correct is 90%

4.2 Assigning credibility to evidence

Since some sources are more suited to provide certain information than others, the choice of sources for the information collection matter a great deal. The credibility of a piece of evidence may be quantified and expressed in a conditional probability matrix. An example is presented in Table 2.

One way of determining the credibility of evidence is to employ heuristics [8]. Such heuristics may state, among other things, that the better the competence profile of the source matches the domain of the question and the more recent the source's information, the higher the credibility. Another way to determine the credibility of evidence is to allow the investigator to specify it subjectively. The elicitation of quantitative estimates from experts is a well-researched field in statistics; see for instance [2] and [12].

In the current example, we estimate the credibility of our source's answers using a simple heuristic, matching their domain of competence with that required by the questions. George has previous experience of the `services`. The investigator therefore estimates George's answers with respect to attributes of the `service` entities to be correct with a 95% probability. Because George is less knowledgeable with respect to the service infrastructure entities (e.g. `service bus`), his credibility is estimated to 90% with respect to the attribute values of those entities. A similar matching between competence profile and model entities is performed for Max. The investigator's own manual inspection of the various artifacts is considered to yield the most credible results, 99% in all cases except for attributes calculated based on other values (e.g. the compatibility of the service bus). In these cases, inspections are slightly better than guesses, 60%. The credibility of the various pieces of evidence is specified in Table 3.

Assuming that the sources do not suffer from bias, the probabilities in Table 3 may be expressed in the form of conditional probability tables, e.g. Table 2.

4.3 Many pieces of evidence

In this simple example, the number of attributes that needs to be assigned values is already quite substantial, twenty-six to be exact. For each attribute there is a possibility to collect three pieces of evidence; in our example this means that there are some seventy-eight pieces of evidence that can be collected. This number will grow rapidly with the addition of new `services`. The addition of each new

		Sources		
Class	Attribute	Max	George	Inspection
Service Cluster	Interoperability	0,6	0,6	0,6
	Service Protocol Compatibility	0,95	0,9	0,6
	Service Bus Compatibility	0,95	0,9	0,6
	Service Orchestration Language Compatibility	0,95	0,9	0,6
Service	Quality	0,6	0,6	0,6
	Correctness	0,9	0,95	0,99
	Availability	0,9	0,95	0,99
	Compatible Service Bus Types	0,95	0,9	0,99
	Compatible Orchestration Languages	0,95	0,9	0,99
Service Interface	Protocol Type	0,95	0,9	0,99
Service Description	Understandability	0,9	0,95	0,99
	Completeness	0,9	0,95	0,99
	Correctness	0,9	0,95	0,99
	Existence in Repository	0,95	0,9	0,99
Service Bus	Type	0,95	0,9	0,99
Service Orchestration Description	Language Type	0,95	0,9	0,99

Table 3
The attributes and the probabilities that the three sources are able to answer questions correctly.

`service` to the cluster will add ten attributes. Assuming ten sources of evidence, this translates to over ten thousand pieces of evidence that could be collected. This reflects the complexity of the problem of service interoperability assessment.

There are two practical problems associated with large concrete models and many pieces of evidence. The first problem is computational. The Bayesian analysis of a large concrete model may be difficult to solve. Generally, the computational requirements grow linearly with the number of cliques and exponentially with the clique size, where a clique is a set of attributes in which every attribute is connected to every other attribute in the set [19]. As a strategy when solving a concrete model, it is possible to maintain the maximum clique size at an acceptable size by introduction of intermediary attributes.

In this paper, we focus on the second practical problem associated with large concrete models and many pieces of evidence. This problem is due to the cost of evidence collection. If every piece of evidence in the service interoperability example would take one minute to collect, it would take a man-month to collect all pieces of evidence, assuming one hundred services and ten sources of evidence. A better option is normally to refrain from collecting all evidence and instead accept a level of uncertainty of the assessment results. When this strategy is employed, it is desirable to collect those pieces of evidence that have the biggest impact on the assessment results to the lowest evidence collection cost. An algorithm to determine which piece of evidence to collect next is described in the next section.

5 Diagnosis in Bayesian networks

To solve the problem of prioritizing pieces of evidence, we turn to the theory of diagnosis in Bayesian networks. Using GeNIe [1], a graphical environment for building decision models created by the Decision Systems Laboratory at the University of Pittsburgh, we then proceed with the demonstration of the example developed above. For simplicity, the following description of diagnosis in Bayesian networks is based upon Jagt's work [6], a master thesis that specifically describes the implementation of the relevant diagnostic functions in GeNIe.

Diagnosis involves two types of tasks: (i) determining of the (combination of) causes of the observed symptoms, and (ii) increasing the credibility of the diagnosis through the collection of additional, initially unobserved, data. Since information seldom comes for free, the second task by necessity involves the formulation of a strategy to gather information as cleverly as possible, i.e. to gain the most diagnostic value at the least cost. We now proceed to make this more precise.

5.1 Value of information

Let a diagnostic probability network (DPN) be defined as a Bayesian network where at least one random variable H is a *hypothesis variable* (in our case the service cluster interoperability) and at least one other random variable T is a *test variable* (in our case the variables of the model that we potentially can collect information about, e.g. George's opinion about the availability of a service).

Let $\mathcal{H}$ denote the set of all hypothesis variables, and $\mathcal{T}$ the set of all test variables. Furthermore, each test $T \in \mathcal{T}$ has a cost function $\text{Cost}(T) : \mathcal{T} \rightarrow \mathbb{R}$. If a test is free, the associated cost is set to zero. Also, each hypothesis H has an associated value function, $V(P(H)) : [0,1] \rightarrow \mathbb{R}$. Given a DPN, we have the expected value EV of performing a test $T \in \mathcal{T}$:

$$EV(T) = \sum_{t \in T} V(P(H \mid t)) \cdot P(t)$$

To make an informed decision, we also need to account for the expected outcome of *not* performing the test T. We therefore introduce the expected benefit EB:

$$EB(T) = EV(T) - V(P(H)) = \sum_{t \in T} V(P(H|t)) \cdot P(t) - V(P(H))$$

Still, however, no connection has been made to the cost of the test. This is remedied by the test strength TS

$$TS(H,T) = \frac{EB(T)}{V(P(H))} - K \cdot \text{Cost}(T)$$

where we have introduced the coefficient K, reflecting the relative importance of the expected benefit versus the cost of the test. The definition of the value function still remains. To optimize the test selection with respect to multiple hypotheses, Jagt introduces a function based on the marginal probability between hypotheses

(rather than the joint probability) called Marginal Strength 1 ($MS1$)

$$MS1(P(F)) \equiv \left(\frac{\sum_{f \in F} (f - 0,5)^2}{0,5^2} - n_F \right) \cdot \frac{1}{n_F}$$

where F is the set of all selected target states f_i of the hypotheses that the user whishes to pursue and nf is the total number of target states. This test selection function is convex with a minimum at $1 - nf$ and maxima at 0 and 1. The value function that we are looking for now becomes the sum of the marginal strength for all target states:

$$V(P(F)) = \sum_{f \in F} MS1(P(f))$$

5.2 Diagnosis in GeNIe

Figure 4 illustrates GeNIe's test ranking user interface. The value function defined above is implemented in GeNIe's multiple cause module ready for use (and is set under the "Options" button). The possible tests are listed to the right, ranked by their diagnostic value as given by the entropy based value function. The entropy/cost ratio set above the list corresponds to the coefficient K defined above.

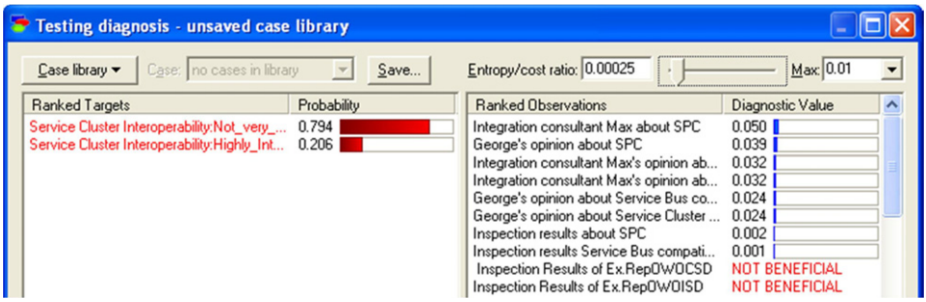

Fig. 4. A ranking of tests in a diagnostic probability network, using GeNIe.

Since there is no standard way of setting this coefficient, it is left to the user to assign an appropriate value. Once a test has been performed it is removed from the list of available tests and the ranking of tests is recalculated and updated to reflect the new situation.

6 Evidence collection costs

The diagnosis algorithm described in the previous section does not only take the value of information obtained from each source into account, it also considers the cost of obtaining the information. In order to use the algorithm to guide data collection for the example presented in section 4, it is thus necessary to assign a cost for each potential source of evidence. The costs assigned here do not have any specific unit like 'dollars'. Rather, they are merely numerical and comparable measures of the effort needed to obtain data from various sources.

It was decided to use consultant Max, system owner George, and the manual inspection (performed by the investigator) as evidence collection methods for all attributes in the model. Since Max the consultant is currently working in the service development project and available for questions most of the time, the cost of retrieving information from Max is fairly inexpensive, the cost was set to 0.5. System owner George, however, is involved in several other projects and is often away on business trips. The cost of interviewing George was therefore set to 1, twice that of interviewing Max. Manual inspection is sometimes an easy and fast method of collecting evidence, e.g. when it comes to determining values of attributes such as service bus type or service description's existence in repository. However, employing manual inspections can also be very time consuming, e.g. when collecting evidence for attributes such as understandability or completeness of a certain service. These types of evidence will require the investigator to really understand the service and its environment, which may be a lengthy process. Therefore, the cost of collecting this evidence with manual inspection is considerable and in the example these were set to 5. See Table 4 for details.

		Source Cost		
Class	Attribute	Max	George	Inspection
Service Cluster	Interoperability	0.5	1	5
	Service Protocol Compatibility	0.5	1	1
	Service Bus Compatibility	0.5	1	1
	Service Orchestration Language Compatibility	0.5	1	1
Service	Quality	0.5	1	1
	Correctness	0.5	1	5
	Availability	0.5	1	5
	Compatible Service Bus Types	0.5	1	0.5
	Compatible Orchestration Languages	0.5	1	0.5
Service Interface	Protocol Type	0.5	1	0.5
Service Description	Understandability	0.5	1	5
	Completeness	0.5	1	5
	Correctness	0.5	1	5
	Existence in Repository	0.5	1	0.5
Service Bus	Type	0.5	1	0.5
Service Orchestration Description	Language Type	0.5	1	0.5

Table 4
Cost table specifying the cost of each source for each possible piece of evidence.

7 Collect data and calculate results

This chapter will demonstrate the usefulness of the diagnosis algorithm for data collection prioritization through a small example. The property of interest is the interoperability of the two `services` 'Work order initiation' and 'Work order closing'. The evidence sources at our disposal are those mentioned above; Max, George and the manual inspection of the various attributes. In this particular case, the `services` and the supporting integration `services` are configured in a manner so as to make them interoperable, so `service cluster` interoperability is in fact 'High'.

The entropy/cost ratio corresponding to the cost coefficient K mentioned above was set to 0.0025. Without having collected any data about the attributes in the concrete model, the conditional probability matrices and the prior probabilities of the concrete model in Figure 3, state that the `service cluster` interoperability is 'High' with a probability of only 21 %. The investigator decides that a credibility level of 90 % for either the 'High' or the 'Low' state is sufficient, and will consequently stop collecting data when this level is reached.

The investigator uses GeNIe's Diagnosis-algorithm to guide the data collection. Using this strategy, the total cost for reaching the targeted 90 % credibility level is about 15, as is shown in Figure 5 below.

It is also evident from Figure 5 that when the investigator does not adhere to any particular strategy when collecting the data the cost of reaching a credibility level of 90 % is on average (based on 10 series) around 56, almost four times higher, a sizeable increase. Faced with a real `service` interoperability analysis involving several hundred `services`, this increase translates into what may be several man-months spent on data collection.

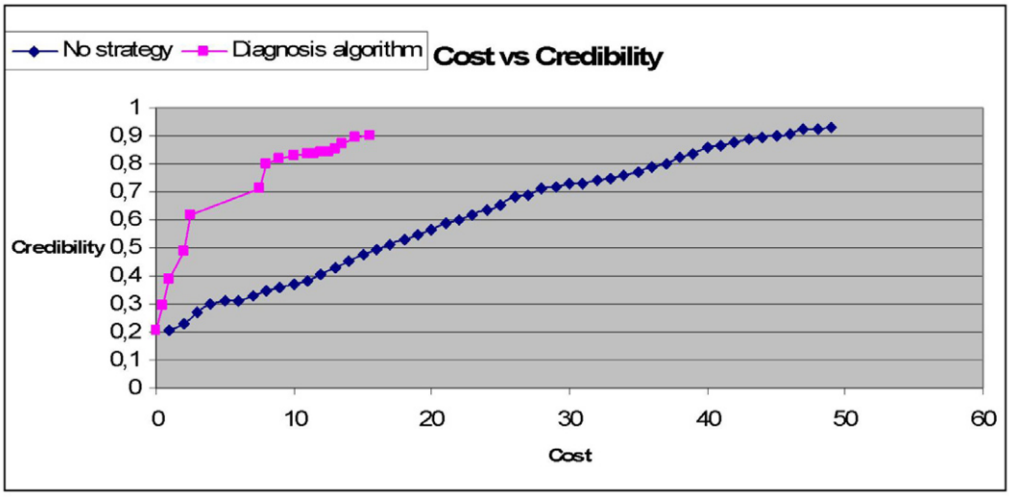

Fig. 5. A comparison of data collection strategies with respect to cost and credibility of results.

8 Conclusions

After having introduced an architectural assessment approach based on Bayesian Networks, this paper has shown that the Diagnosis Algorithm, as implemented in the GeNIe-tool can be used as support when wanting to minimize the time spent on collecting data during investigations of complex software qualities such as service interoperability.

An example, which was implemented in GeNIe, featured the analysis the interoperability of a very small cluster of services. In the example it was demonstrated that the use of the diagnosis algorithm reduced the cost of data collection by almost as compared to collecting data without the use of any particular strategy. From this we conclude that the diagnosis algorithm offers a possibility of lowering the cost of performing system quality analyses quite significantly.

References

[1] Decision System Laboratories, "About GeNIe and SMILE", University of Pittsburgh, 2008, URL: http://genie.sis.pitt.edu/about.html

[2] Druzdzel M. and L. van der Gaag, *Building probabilistic networks: Where do the numbers come from?*, IEEE Transactions on knowledge and data engineering, **12**, IEEE, Los Angeles, USA, 2000, 289 - 299

[3] Erl T., "Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services", Prentice Hall, New Jersey, 2004

[4] Erl T. "Service-Oriented Architecture: Concepts, Technology, and Design" Prentice Hall, New Jersey, 2005

[5] Friedman N., M. Linial, I. Nachman, and D Pe'er, *Using Bayesian Networks to Analyze Expression Data*, Journal of Computational Biology, textbf7, Mary Ann Liebert, Inc., New Rochelle, NY, 2000, 600-620

[6] Jagt, R. M, "Support for Multiple Cause Diagnosis with Bayesian Networks", unpublished M.Sc. Thesis, Department of Mediamatics, Information Technology and Systems, Delft University of Technology, the Netherlands and Information Sciences Department, University of Pittsburgh, Pitsburgh, PA., 2002

[7] Jensen F. V., "Bayesian Networks and Decision Graphs", Springer New York, Secaucus, NJ, USA, 2001

[8] Johansson E. and P. Johnson, *Assessment of Enterprise Information Security - Estimating the Credibility of the Results*, Proceeding of the Symposium on Requirements Engineering for Information Security (SREIS) in the 13th International IEEE Requirements Engineering Conference, **13** Paris, France, 2005

[9] Johnson P., R. Lagerström, P. Närman and Mårten Simonsson, *Enterprise architecture analysis with extended influence diagrams*, Information System Frontiers, **9**, Springer Verlag, The Netherlands, 2007, 163-180

[10] Johnson P., R. Lagerström, P. Närman and M. Simonsson, *Extended Influence Diagrams for System Quality Analysis*, Journal of Software, **2**, Academypublisher, 2007, 30-42

[11] Johnson P., Johansson E., T. Sommestad and J. Ullberg, *A Tool for Enterprise Architecture Analysis*, Proceedings of the 11th IEEE International Enterprise Distributed Object Conference (EDOC),**11** IEEE Computer Society, Annapolis, October 2007, 142-152

[12] Keeney R. and D. von Winterfeldt, *Eliciting Probabilities from Experts in Complex Technical Problems*, IEEE Transactions on engineering management, **38**, IEEE, 1991, 191-201

[13] Marks E., M. Bell, "Service-Oriented Architecture: A Planning and Implementing Guide for Business and Technology", John Wiley & Sons, New Jersey 2006

[14] Neapolitan R., "Learning Bayesian Networks", Prentice-Hall Inc. Upper Saddle River, NJ, USA 2003

[15] The Object Management Group (OMG, "Unified Modeling Language: Superstructure. version 2.1.1, 2007, URL: http://www.omg.org/technology/documents/formal/uml.htm

[16] D. Georgakopoulos and , Papazoglou M. *Service-Oriented Computing*, Communications of the ACM, **46**, ACM, 2003, 25-28

[17] Papazoglou M., *Service-Oriented Computing: Concepts, Characteristics and Directions*, Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03), textbf4 IEEE, 2003, 3-12

[18] Pearl J., "Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, San Mateo, 1988

[19] Rish I. and R. Dechter, *On the impact of causal independence*, Stanford Spring Symposium on Interactive and Mixed-Initiative Decision Theoretic Systems, Stanford, 1998

[20] Shachter R., *Probabilistic inference and influence diagrams*, Operations Research, **36**, Institute for Operations Research and the Management Sciences, 1988, 589-604

[21] Ullberg J., R. Lagerström and Johnson P., *Enterprise Architecture: A Service Interoperability Analysis Framework*, Proceedings of the 4th International Interoperability for Enterprise Software and Applications Conference (I-ESA'08), **4**, Springer Verlag, Berlin, 2008, 611-623