Extended Influence Diagrams for System Quality Analysis

Pontus Johnson, Robert Lagerström, Per Närman, Mårten Simonsson
Department of Industrial Information and Control Systems, Royal Institute of Technology, Stockholm, Sweden
Email: {pj101, robertl, pern, martens}@ics.kth.se

Abstract— **Making major changes in enterprise information systems, such as large IT-investments, often have a significant impact on business operations. Moreover, when deliberating which IT-changes to make, the consequences of choosing a certain scenario may be difficult to grasp. One way to ascertain the quality of IT investment decisions is through the use of methods from decision theory. This paper proposes the use of one such method to facilitate IT-investment decision making, viz. extended influence diagrams. An extended influence diagram is a tool able to completely describe and analyse a decision situation. The applicability of extended influence diagrams is demonstrated at the end of the paper by using an extended influence diagram in combination with the ISO/IEC 9126 software quality characteristics and metrics as means to assist a decision maker in a decision regarding an IT-investment.**

Index Terms— **extended influence diagrams, system quality analysis, IT-investment decision making, ISO/IEC 9126**

I. Introduction

Contemporary companies rely on the use of IT to conduct business operations. Every once in a while all companies are forced to make decisions regarding large IT-investments. The reason for this could for instance be the need for replacing legacy systems. In addition to this, what used to be stand-alone, stovepipe IT-systems are increasingly becoming integrated into an enterprise-wide system of systems. Thus, a decision made concerning local a change in the enterprise architecture may have an adverse impact on a whole range of other systems. The importance and complexity of IT-systems call for a structured approach to the making of decisions concerning IT-investments.

The discipline of decision theory [1] is concerned with the formal specification of decision problems, and provides tools facilitating stringent decision analyses. Using decision theory can, if employed correctly, improve the quality of decisions. In [2], [3], and [4] an attempt is made to apply theoretical decision methods on IT-related decision analysis. So called *extended influence diagrams* are introduced and applied for architectural analysis. Extended influence diagrams are extensions of an existing decision tool, *influence diagrams* [5] [6] [7] [8] [9], which are commonly employed in decision analysis.

IT-investment situations frequently involve the choice of one of several candidate system scenarios. Apart from analyzing the cost of the scenarios, an important factor to consider is the software quality of the respective system scenario. To facilitate such evaluations ISO/IEC has developed and presented software quality characteristics and metrics in the ISO/IEC 9126 International Standard [10] and Technical Reports [11] [12] [13]. Using ISO 9126 is a way to define the term ”software quality”, and to specify how it may be measured. This paper proposes the use of extended influence diagrams together with ISO 9126 characteristics, subcharacteristics and metrics to assess system scenarios before making an IT-investment decision.

The remainder of this paper is outlined as follows. In section 2 we elaborate on decision problems in general, and IT-investment decisions in particular. In section 3 and 4 we aim at giving a rich description of extended influence diagrams and how they differ from normal influence diagrams. Section 5 briefly summarizes how an extended influence diagram may be constructed in a decision situation, and section 6 and 7 demonstrate how an extended influence diagram can be used to assist a decision maker by using the ISO 9126 framework for assesing the software quality of different IT investment candidates. In section 8 we conclude the article. Sections 2 through 5 are similar to our material presented in [2], [3], and [4].

II. System quality analysis

A rational decision maker is an agent facing an alternative after a process of deliberation in which he or she answers three questions: “What is feasible?”, “What is desirable?” and “What is the best alternative according to the notion of desirability, given the feasibility constraint?” [1] [14]. Translating these questions into the context of system quality analysis, change scenarios answer the first question of what is feasible. Answers to the second question regarding desirables, are often expressed in terms of the change of various information system qualities such as increased information security, increased interoperability, increased availability, etc. The answer to the third question, providing the link between the feasible to the desirable, i.e. the link between the scenarios and the properties of interest, is given by system quality analysis.

This paper is based on “System Quality Analysis with Extended Influence Diagrams,” by P. Johnson, R. Lagerström, P. Närman and M. Simonsson, which appeared in the Proceedings of the 11th IEEE Conference on Software Maintenance and Reenginering (CSMR) - Special Session on System Quality and Maintainability (SQM), Amsterdam, the Netherlands, March 2007. © 2007 IEEE.

Decision making, whether the decisions apply to IT or not, is rarely performed under conditions of complete certainty. One fundamental uncertainty is regarding the definition of various concepts. This is called *definitional uncertainty*. For instance, some authors define the term information security in terms of confidentiality, integrity and availability, [15] whereas others add the concepts of non-repudiation and accounting to the definition [16].

Related to definitional uncertainty is *theoretical heterogeneity*. Existing knowledge regarding the nature of enterprise information systems is not consolidated within a single commonly accepted framework, as may be the case for more mature disciplines. A consequence of this is that there is a need to relate similar concepts to each other, and to relate concepts at varying levels of abstraction to each other.

In addition to definitional uncertainty, there may also be uncertainty with regard to how the world actually behaves. Knowledge of exactly how various phenomena affect one another is seldom certain. There is a level of *causal uncertainty*. An example of this would be uncertainty with respect to the causal effect the percentage of systems with updated virus protection may have on the level of information security.

When analyzing systems of an enterprise, the information represented in the analysis is normally associated with a degree of uncertainty. Perhaps the information was collected a while ago and has now become obsolete, or perhaps the information was gathered from a source that might have been incorrect. In these situations, the decision maker suffers from *empirical uncertainty*.

From this description of the context of system quality analysis, requirements may be extracted on the languages used for analysis specification. As is demonstrated in [2] and [3] extended influence diagrams, which are an augmented version of influence diagrams [7] fulfill all such requirements. This paper proceeds to explain what influence diagrams are, and the nature of the extension that makes influence diagrams into extended influence diagrams. An example on the use of extented influence diagrams for IT decision-making is also presented.

III. Influence diagrams

This section briefly describes conventional influence diagrams. For a more comprehensive treatment, the reader is referred to [5], [6], [7], [8], and [9].

An *influence diagram* is an extension of Bayesian networks. A Bayesian network graphically represents causal relations between nodes, where each node represents a variable with a number of states. Moreover, Bayesian networks are able to represent the uncertainty of the causal relations using probabilistic reasoning. Using the terminology from section II, a Bayesian network is able to answer the question ”what is feasible?”, through the modelling of the real world. An example of a Bayesian network is shown in Figure 1. The example shows that there is a causal dependence between the variable ”time spent studying mathematics”, and the variable ”understanding of mathematics” and the direction the arrow points suggests that the former affects the latter, rather than vice versa.

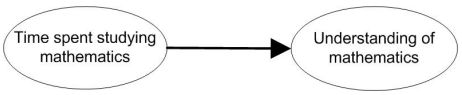

Figure 1. A very simple Bayesian network showing the causal dependence between time spent studying mathematics and understanding of mathematics.

To capture that most relations are probabilistic rather than deterministic in nature - in some cases a student could spend a considerable time studying and still not get a better understanding of the subject - Bayesian networks use so called *conditional probability matrices*. A conditional probability matrix captures the likelihood of a variable being in a state X, under the condition that it’s ”predecessor” is in a state Y. See Figure 2.

Time spent studying		5 hours	3 hours	1 hour	0 hours
Understanding of Mathematics	Good	0.8	0.5	0.2	0.05
	Passable	0.15	0.4	0.2	0.15
	None whatsoever	0.05	0.1	0.6	0.8

Figure 2. A conditional probability matrix showing the probability of a student’s understanding of mathematics given the time spent studying. In this case it is very *probable* (80%) that a student will have a good understanding if he or she spends 5 hours studying, but it is not certain.

In addition to representing causal relations, influence diagrams support a more complete and intuitive description of decision problems, stating both what is desired, and what alternatives are available. This is accomplished by the introduction of two new nodes. The first is the *decision node*, which represents the decision alternatives at hand. The second is the *utility node* where the outcome of a decision is quantitatively assessed as an expected utility. To illustrate, we expand the example in Figure 1 to include also the decision node ”Decision to study” wherein the student decides whether to study mathematics or not. Also included is the utility node ”Mathematics grade” which in this particular case is the variable that the student wishes to maximize. In this simple example, the decision analysis is straightforward and suggests that the student ought to decide to study. See Figure 3.

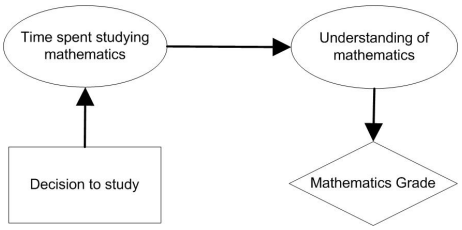

Figure 3. A very simple influence diagram showing not only the causal dependence between time spent studying mathematics and understanding mathematics, but also detailing the student’s goals and alternatives.

More rigorously, an influence diagram is a network used for modelling uncertain variables and decisions, consisting of a directed graph $G = (N, A)$. There are three types of nodes in the set N, partitioned into the sets V, C and D. There are one or more *utility* nodes $v \in V$, these are depicted as rhombuses. There are zero or more *chance* nodes $c \in C$, these are depicted as ovals. There are one or more *decision* nodes $d \in D$, and these are depicted as squares. See Figure 4.

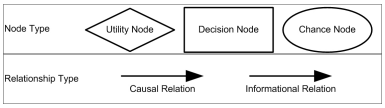

Figure 4. Utility, chance, and decision nodes in influence diagrams.

There are two types of arcs in the set A, partitioned into the sets K and I. Arcs into utility and chance nodes are *causal*, $\kappa \in K$, representing probabilistic dependence. Arcs into decision nodes are informational, $\iota \in I$, and imply time precedence. Both the causal relation arcs and the informational relation arcs are depicted as arrows. See Figure 4.

The set of causal predecessors are represented by $CP(i) = \{j \in N : (j, i) \in K\}$, where $i \in \{V, C\}$. The set of informational predecessors are represented by $IP(i) = \{j \in N : (j, i) \in I\}$, where $i \in D$.

Associated with each node i is a variable X_i and a set Ω_i of possible values it may assume. If i is the utility node, then X_i represents the expected utility and its domain Ω_i is a subset of the real line. If i is a chance node, then Ω_i is the sample space for the random variable X_i. Finally, decision node i has alternative X_i chosen from the set Ω_i. The utility node $v \in V$ has an associated utility function $U : \Omega_{CP(v)} \rightarrow \Omega_v$, which represents the expected utility as a function of the values of the conditioning predecessors of the utility node. There is a conditional probability distribution, Pr, for every chance node i, given the values of its causal predecessors, $Pr\{x_i|x_{CP(i)}\}$. The notation $\Pr\{x_i|x_{CP(i)}\}$ means that $\Pr\{X_i = x_i|X_{CP(i)} = x_{CP(i)}\}$.

The probability distributions for the utility and each chance node are represented in conditional probability matrices. Figure 5 shows the conditional probability matrix for a chance node y dependent on a node z.

Z		z1	z2
Y	y1	Pr(y1|z1)	Pr(y1|z2)
	y2	Pr(y2|z1)	Pr(y2|z2)

Figure 5. A conditional probability matrix.

IV. Extended influence diagrams

Although influence diagrams may be used for system quality analysis in their conventional form, there are some requirements that are not sufficiently addressed. This section therefore presents a set of extensions to influence diagrams for the purposes of system analysis.

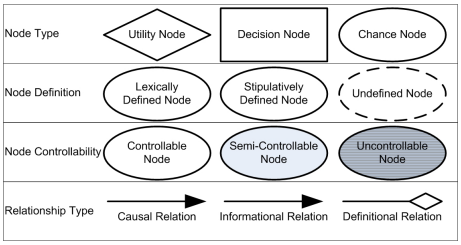

Figure 6. The syntax of extended influence diagrams.

A. Lexically defined nodes

Causality means that one phenomenon in the real world somehow affects another. Because causality is a concept of the real world, it is important that there is a mapping between the real world and the influence diagrams, i.e. that the concepts presented in the influence diagrams are well-defined. If they are not well-defined, it will be impossible to determine whether there is in fact any truth to the causal relation between the phenomena.

The definitions of some nodes in influence diagrams are deemed uncontroversial. For instance, in the IT community, we might assume that there is common agreement on the definition of the concept of memory size; it is typically measured in terms of bytes. Nodes that we consider uncontroversially defined are called *lexically defined nodes*, $L \subset N$ [17]. Figure 6 details the syntax of extended influence diagrams.

B. Stipulatively defined and undefined nodes

In the considered decision-making context, we have assumed that there is considerable confusion as to the meaning of many concepts, such as information security. There are thus many potential nodes that are not lexically defined. In order to manage these, we introduce the possibility to define nodes within the influence diagram. This is done by relating a node, directly or indirectly, to lexically defined nodes with a new kind of arc, called a definitional relation, $\delta \in \Delta$, where $\Delta \subset A$, see Figure 6. Nodes that are defined in the diagram using the definitional relation are called *stipulatively defined nodes* $S \subset N$ [17]. Those nodes that are neither lexically nor stipulatively defined are simply *undefined*, $U \subset N$.

C. Definitional relations

This subsection details the semantics of definitional relations, Δ. The set of defining nodes is given by $DF(i) = \{j \in N : (j, i) \in \Delta\}$ where $i \in \{V, C\}$. The definitional relation implies simple aggregation; the defined node is comprised of its constituent parts. The definitional relation is represented mathematically by the same conditional probability distributions as are used for causal relations, $Pr\{x_i|x_{DF(i)}\}$. Since definitional relations are simple

aggregations, a node that has defining nodes cannot also in the same graph feature causal predecessors.

The concept of definitional relations is similar to the aggregation mechanism provided by object-oriented influence diagrams [18]. However, aggregated objects in object-oriented influence diagrams are to be viewed more as placeholders for diagrams than as nodes. They are therefore not associated with any variable X_i. As is considered below, the association of a variable with the aggregated node is of significant importance for our purposes.

D. Controllability of nodes

A utility node may be affected by many chance nodes, and it may be the case that only some of these chance nodes are affected (directly or indirectly) by the decision node. Those nodes that lie in a causal path from decision node to utility node or are stipulatively defined by such nodes are relevant for decision making. Other nodes are not. Nodes that are completely dependent on the decision node are denoted *controllable nodes*, $F \subset N$. Nodes that are completely independent of the decision nodes are denoted *uncontrollable nodes*, $W \subset N$. Finally, nodes that are partially dependent on the decision node are denoted *semi-controllable nodes*, $SC \subset N$. The graphical rendering of these node types is given in Figure 6.

Furthermore, it may sometimes be useful to distinguish between *directly controllable* nodes, $DC \subset F$ and *indirectly controllable nodes*, $IDC \subset F$. Directly controllable nodes are targets of a causal relation originating in a decision node, $DC(i) = \{j \in D : (j,i) \in K\}$. Indirectly controllable nodes are targets of a causal relation originating in a directly controllable node or an indirectly controllable node, $IDC(i) = \{j \in \{DC, IDC\} : (j,i) \in \{K, \Delta\}\}$.

V. Extended influence diagrams for system quality analysis

Recalling the decision making approach described in section II, the question "What is desirable?" may be specified by an extended influence diagram utility node, $v \in V$ (see Figure 7). An example of an utility node concept for system quality analysis is *Information Security*.

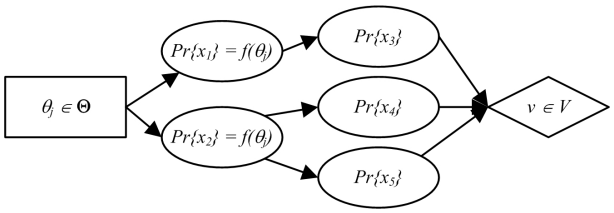

Figure 7. The relation between scenario selection, content of the scenarios, intermediary chance nodes, and the utility node.

The answer to "What is feasible?" is in the case of system quality based decision making answered by a set of change scenarios,$\Theta = \{\theta_1, ...\theta_n\}$. The characteristics of such scenarios may then be used to determine the conditional probability distributions of chance nodes in the extended influence diagram, $Pr\{x_i\} = f(\theta_i)$. The decision maker's choice between scenarios is represented by a decision node, $\theta \in \Theta$, which is connected by causal relations to all those chance nodes that the scenario selection might affect.

The chance nodes that are directly affected by the decision node also need to be related to the utility node so that the utility of the different scenarios can be calculated. Typically, it is performed by the use of intermediary chance nodes. The resulting influence diagram is a representation of how we believe that the real world functions; it is the representation of a theory of, for instance, information security. The diagram thereby provides the answer to the decision maker's third question, "What is the best alternative according to the notion of desirability, given the feasibility constraint?"

The rest of this chapter describes a generic process for construction of extended influence diagrams and the use of them for system quality analysis. The construction process is described in section V-A while the process of system quality analysis is described in section V-B.

A. Development of extended influence diagrams

Figure 8 depicts a process for the construction of extended influence diagrams. This subsection describes that process step by step. The focus of this article is on the conceptual rather than practical aspects of the generation process. For practical techniques for the elicitation of extended influence diagrams, the reader is referred to [19].

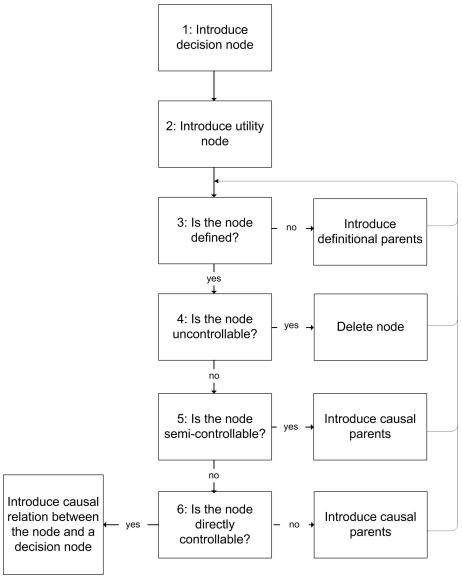

Figure 8. The process of developing extended influence diagrams.

1) Step one: Introduce decision node: At the very start, the decision node should be identified. As mentioned above decision nodes represent the choice between different change scenarios, such as the choice between integrating a set of systems, replacing them, or maintaining the status quo.

2) Step two: Introduce utility node: The second step is to define the utility node. The utility node is the target of the system quality analysis. Examples of utility nodes are *information security*, *modifiability*, *performance* and *reliability*. When defining the utility node, its variable type should also be decided on, e.g. $\{Low, Medium, High\}$, $\{Present, Absent\}$, $\{True, False\}$, or $\{0, 1, 2, 3, 4\}$.

3) Step three: Is the node defined?: The third step is to determine whether the node is lexically defined, stipulatively defined or undefined. Recall that a node is lexically defined if we can assume that there is common agreement on its definition. Although almost all definitions can be challenged, concepts that would normally qualify as lexically defined include *weight in kilograms*, *number of processors* or *number of users*. For many concepts, however, such as *information security*, *architecture quality* and *competence* there is no universal definition even by very pragmatic standards.

If the node is not lexically defined, it needs to be stipulatively defined. This is accomplished with the definitional relationship presented in the previous chapter. As an example, the node *Availability* might be defined by the two nodes *Mean Time To Failure* and *Mean Time To Repair*. The stipulative definition of a node results in the introduction of new nodes into the diagram. When new nodes are introduced, this affects the conditional probability matrices of the child nodes. These must therefore also be specified, thereby detailing the dependencies between the parents and the child. When an undefined node has been stipulatively defined, the diagram construction process refocuses on one of the new nodes and returns to the third step.

4) Step four: Is the node uncontrollable?: For each defined node, the fourth step considers whether the node is uncontrollable. Recall that uncontrollable nodes are unaffected by variable changes in the decision node; this means that potential variations in the value of the uncontrollable node are unrelated to the choices of the decision maker. If the node is uncontrollable, it will not provide decision supporting information, so there is no need to continue exploring this branch of the diagram. The node is therefore deleted, the process refocuses on the next unexamined node and returns to the third step.

5) Step five: Is the node semi-controllable?: Step five queries whether the node under consideration is semi-controllable. Semi-controllable nodes are affected both by the decision makers choices and other, uncontrollable, phenomena. If a node is semi-controllable, it is important to separate those aspects which are controllable from those which are not. Therefore, new nodes are introduced into the diagram with causal relations to the semi-controllable node under consideration. As an example, we might believe that the semi-controllable node *Mean Time To Repair* is causally affected by both the *Maintainability* of the system and *Flexibility of Working Hour Regulations*. Of these, the maintainability might be controllable, while the working hour regulations might be beyond the decision makers domain of control. In the same manner as in the second step, the involved conditional probability matrices need to be specified. The diagram construction process once again refocuses on one of the new nodes and returns to the third step.

6) Step six: Is the node directly controllable?: In the sixth step, the process considers whether the nodes are directly or indirectly controllable. A directly controllable phenomenon is an immediate consequence of the decision makers choice. For instance, if a scenario is chosen where one system is replaced by another, this may directly entail that the CPU speed is increased. Directly controllable nodes are causally connected to the decision node. For nodes that do not seem to be directly controllable, one or several new nodes are introduced into the diagram, the focus shifts to the first of these, and the process returns to the third step.

When the whole process is finished, the result is an extended influence diagram where the nodes are causally affected by the decision node and in turn either causally affect the utility node, or do so by definition. No nodes are undefined, and uncontrollable nodes have no parents.

B. Analyzing with extended influence diagrams

When an extended influence diagram has been constructed, it may be used to compare and assess the quality of different system scenarios.

1) Linking scenarios to extended influence diagrams: In extended influence diagrams, decision nodes represent a choice between alternatives. In system quality analysis, these alternatives are concretized by different change scenarios. We will now consider how the information represented in the scenarios is introduced into the extended influence diagrams.

A change scenario, say *Scenario X*, can contain a set of entities. Considering one of these entities, say the *System A* entity, we find that it features a set of attributes, *Memory Size*, *Lines Of Code*, etc. The value of each of these attributes is represented in a conditional probability matrix, $Pr\{MemorySize_{EA} = ms_i\}$. This approach retains the possibility to present attribute values deterministically by allowing only zero or unity probabilities in the matrix.

The directly controllable nodes in a extended influence diagram, i.e. the chance nodes that are directly linked to the decision node, consitute the coupling to the scenario. Examining one of these nodes in detail, we find that it might be named *Memory Size*. The link between the scenario and the extended influence diagram is concretely specified by the following requirement: the *Memory Size* node's probability given that *Scenario X* is selected is equal to the conditional probability of the *Memory Size* attribute of the *System A* entity, i.e. $Pr\{MemorySize_{EID} = ms_i | d_{ScenarioX}\} = Pr\{MemorySize_{EA} = ms_i\}$.

2) Calculating the results: The conditional probability distributions of the directly controllable nodes are thus retrieved from the change scenarios. The higher-level conditional probability matrices were determined already during the extended influence diagram construction process.

The value of the utility node can therefore be calculated employing standard methods [8] [9]. There are several tools available on the market for these calculations, such as Hugin [18] and Genie [20].

VI. The ISO/IEC 9126 extended influence diagram

A. Background to ISO/IEC 9126

Much has been written on the topic of software quality measurement [21] [22]. During the seventies a few models for software quality assessment were proposed by McCall [23], Boehm [24] and Grady [25]. All of these models are so called factor-criteria models, referring to the hierarchical structure of the models. McCalls model was developed by the US Air Force and General Electrics, and has since been used for software quality assessments predominantly in the space and avionics field. Drawing heavily on Boehm's and McCall's work, the ISO/IEC's JTC1 initiated a project to consolidate software quality assessment frameworks into one coherent body of knowledge.

The resulting documents ISO 9126-1 [10], 9126-2 [11], 9126-3 [12], and 9126-4 [13], of which only the first is a standard, divide software quality measurements into three parts. Firstly, internal quality measurements, detailed in ISO 9126-3, refer to assessments of the internal behaviour of the software quality product. The software is viewed as a white-box entity, and the metrics are consequently on a low level of abstraction. The primary use of such a model is to simplify software quality assessment during the software development process. Secondly, ISO 9126-2 elaborates on the external quality part of ISO 9126. It focuses on software products post development by taking the viewpoint of a system end-user. External metrics are suitable whenever a software product has been developed and should be assessed for the purpose of for instance testing. The third and last part of the standard, primarily dwelt upon in ISO 9126-4 measures the software's quality in use, thereby referring to the software's effect on the business environment it is implemented in rather than the software product per se. The categorization does not imply that the metrics are disconnected from one another. On the contrary, the idea is that there is the quality in use metrics are causally dependant on the external metrics, which in turn are supervenient on the internal metrics and the internal behaviour of the software.

The internal and external metrics are convenient for use in this paper since what is to be evaluated is the difference in quality between a number of scenarios involving changing an existing software product installation with respect to a number of parameters. What is desired are some kind of pre-implementation indications of the quality in use metrics. The best way of doing so is through the use of internal and external metrics as described in ISO 9126-2 and ISO 9126-3.

B. External and internal quality of software products

In this section, an extended influence diagram over the ISO/IEC 9126 is presented. As mentioned, this standard is particularly suitable for these purposes since it proposes a quality model for software products. The presented influence diagram focuses on the external and internal quality, since the third part, quality in use, is affected by many system-external factors.

Following the extended influence diagram development process presented in section V-A, the first node to introduce is the decision node (cf. Figure 9). As mentioned previously, the relevant decision situation in this case is between a set of different IT investment scenarios, for instance between Software Product X and Software Product Y. The next step in the development process is to define the utility node. It is clear from the standard that the utility node in this case should be named External and Internal Quality. Since the ISO/IEC 9126 does not specify the domain of this variable, we assign to it the set high, medium, low.

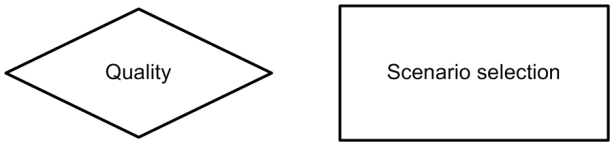

Figure 9. Figure over decision and utility nodes.

1) Defining top node by six software quality characteristics: According to the influence diagram generation process, this node needs to be defined. ISO/IEC 9126 informs us that there are six software quality characteristics, namely functionality, reliability, usability, efficiency, maintainability, and portability (figure 10). The functionality is described as the capability of the software product to provide functions which meet both stated and implied needs. The reliability is described as the capability of the software product to maintain a specified level of performance. Usability is the capability of the software product to be understood, learned, used and also attractive to the user. Efficiency is the capability of the software product to provide appropriate performance, relative to the amount of resources used. Maintainability is the capability of the software to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in the environment, and in requirements and functional specifications. Finally, portability is the capability of the software to be transferred from one environment to another. These six nodes are assigned the same high, medium, low domain as the top-level node, external and internal quality.

The relation between the top-level node and the six defining nodes must be specified in a conditional probability matrix. Since the ISO/IEC 9126 does not detail this relation, we assume it to be an unweighted average. When all defining nodes assume the value high, the top-level node also assumes this value. When three defining nodes

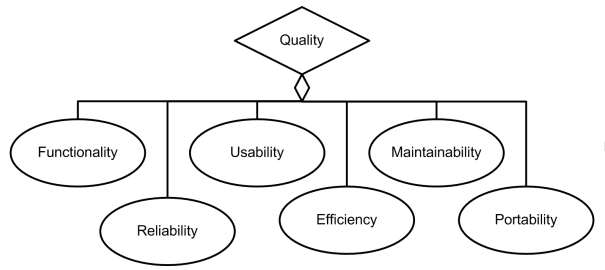

Figure 10. Extended influence diagram over top-level nodes of ISO/IEC 9126.

are high and three are low, the top-level node assumes the value medium.

2) Defining quality characteristics by subcharacteristics: The six software quality characteristics are still very abstract and difficult to measure directly. It is therefore necessary to define them stipulatively in terms of less abstract concepts. Functionality is therefore defined in terms of suitability, accuracy, interoperability and security (cf. Figure 11). Suitability describes the capability of the software product to provide an appropriate set of functions for specified tasks and user objectives. Accuracy is the capability to provide the right or agreed results or effects with the needed degree of precision. Interoperability is described as the ability to interact with one or more specified systems. Security is the capability to protect information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them. All the six software quality characteristics are also measured in terms of their compliance, i.e. their capability to adhere to standards, conventions or regulations relating to the various characteristics. This node will therefore reappear for all of the characteristics below.

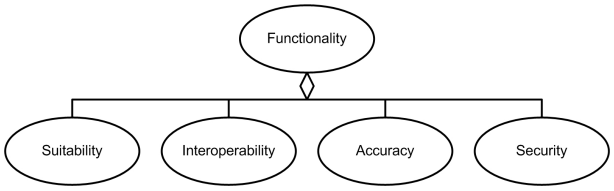

Figure 11. Extended influence diagram over functionality.

Reliability is defined in terms of the product's maturity, fault tolerance and recoverability (cf. Figure 12). Maturity is the capability to avoid failure as a result if faults in the software. Fault tolerance is the capability to maintain a specified level of performance in cases of software faults or of infringements of its specified interfaces. Recoverability is the capability to re-establish a specified level of performance and recover the data directly affected in the case of failure.

Usability is defined in terms of the products understandability, learnability, operability and attractiveness (cf. Figure 13). Understandability relates to the capability to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and

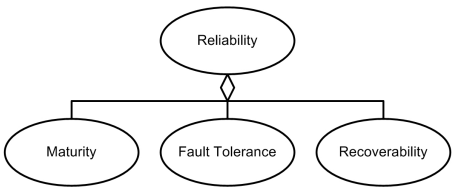

Figure 12. Extended influence diagram over reliability.

conditions of use. Learnability enables the user to learn the software's application. Operability enables the user to operate and control the software. Attractiveness concerns the user's perception of the software.

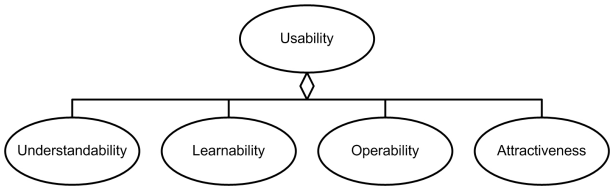

Figure 13. Extended influence diagram over usability.

Considering efficiency, this characteristic is defined in terms of time behavior and resource utilization (cf. Figure 14). Time behavior is the capability to provide appropriate response and processing times and throughput rates when performing a function while resource utilisation is the capability to use appropriate amounts and types of resources when performing the functions.

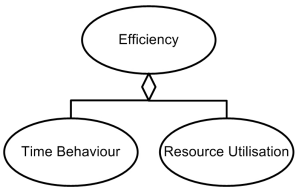

Figure 14. Extended influence diagram over efficiency.

A software product's maintainability is defined by its analyzability, changeability, stability and testability (cf. Figure 15). Analysability is the capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified. Changeability is the capability to enable a specified modification to be implemented. Stability is the capability to avoid unexpected effects from modifications of the software. The testability of a modified software product is its capability to be validated.

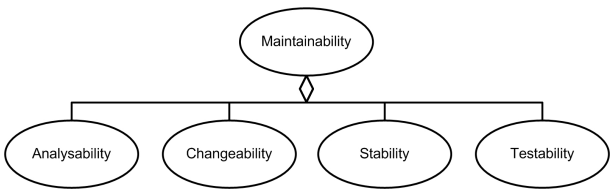

Figure 15. Extended influence diagram over maintainability.

Finally, portability is defined in terms of the adapt-

ability, installability, co-existence and replaceability of the software (cf. Figure 16). Adaptability is the capability of the software to be adapted for different specific environments without applying actions or means other than those provided for this purpose for the software considered. Installability is the capability of the product to be installed in a specified environment. Co-existence is the capability of the software to co-exist with other independent software in a common environment sharing common resources. Replaceability is the capability of the product to be used in place of another specified software product for the same purpose in the same environment.

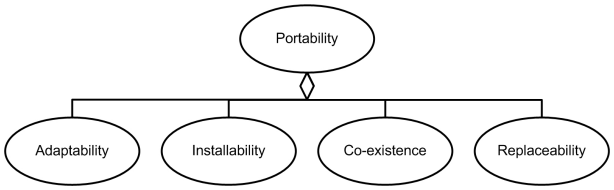

Figure 16. Extended influence diagram over portability.

Each of these software quality characteristics is related to its defining nodes with a conditional probability matrix of the same kind as for the case of the top-level node.

3) Measuring the software quality subcharacteristics: The definition of the quality characteristics in terms of subcharacteristics have increased the degree of concreteness, but the subcharacteristics are still undefined. ISO/IEC 9126-1 does not provide further stipulative definitions, so it is not possible to continue the breakdown. However, in two technical reports related to the standard, ISO/IEC 9126-2 and ISO/IEC 9126-3 a set of metrics are suggested for each subcharacteristic. The reports emphasize that the proposed set of metrics is incomplete. In this article, we will use the metrics for the subcharacteristics interoperability and security to demonstrate how these metrics are represented in the extended influence diagram.

Three metrics are proposed for interoperability (figure 17). The first is data exchangeability (data format based), which measures the percentage of correct interface data formats exchanged with other systems. The second metric is data exchangeability (user's success attempt based), which measures the percentage of users' attempts to exchange data that were successful. The third metric is interface consistency (protocol), measuring the share of correctly implemented interface protocols as compared to the total required number of protocols.

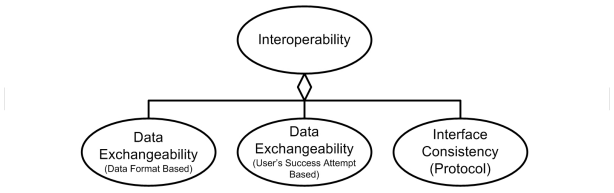

Figure 17. Extended influence diagram over interoperability.

Four security metrics are proposed (figure 18). The first of these is the access auditability, specifying the number of logged access types as compared to the logging requirements. The second security metric is access controllability, measuring the percentage of illegal operations that are detected. The third metric is data corruption prevention, measuring the share of data corruption attempts that were successful in a given test situation. The fourth metric is data encryption, counting the percentage of sensitive data items that are encrypted. The conditional probability matrices for the metrics are of the same kind as for the quality characteristics.

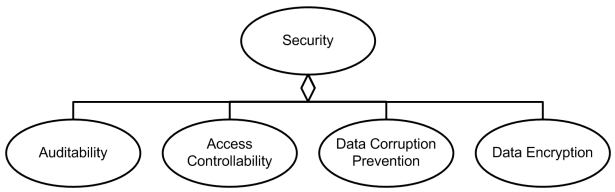

Figure 18. Extended influence diagram over security.

The metrics are not only well-defined but they also depend directly on the choice of scenarios. Therefore, the decision node introduced earlier is linked to the metrics as a causal parent.

VII. APPLYING THE ISO/IEC 9126 EXTENDED INFLUENCE DIAGRAM FOR QUALITY ANALYSIS OF A COLLABORATION SYSTEM

In order to show the applicability of the proposed extended influence diagram, this section provides an example on the use of extended influence diagrams for analysis of the quality of a collaboration system employed at a university.

A. The collaboration system

The collaboration system was originally created to allow teachers to communicate with their students, by for instance displaying assignment results and inform students in case a lecture was rescheduled. The collaboration system initially had limited functionality, but due to new user requirements, the system has grown considerably over the past few years. Today, the system aids in communication between employees and students by chat functionality, file sharing for peer collaboration, assignment management, and support for scheduling of lectures and workshops. Overall the collaboration system works well. However, the system is a result of a small group of people's programming effort distributed over many years. The system was developed in an ad hoc manner and the development process did not follow any standardized approach. This has not only reduced the system's maintainability, but also made it less interoperable with other systems. Some questions regarding the safeguarding of information had also been raised recently and an encrypted data storage would be highly desireable.

Due to an ongoing IT system cleanup project, the IT department of the university considered whether the collaboration system should be further expanded or not.

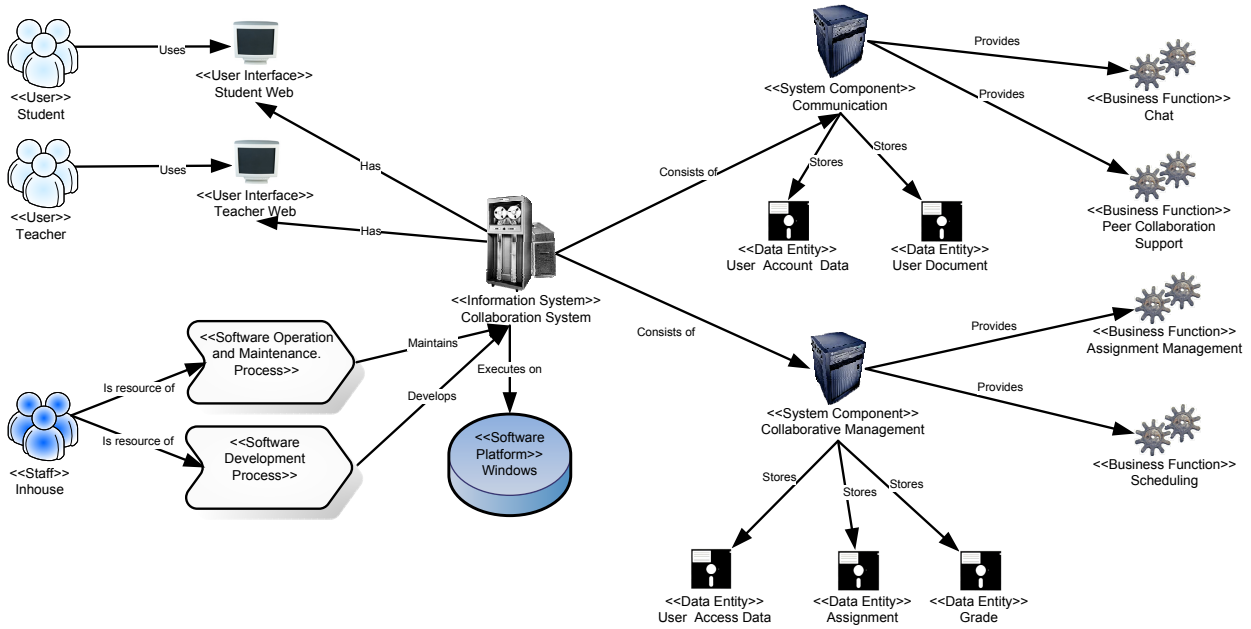

Figure 19. Scenario 1 for the collaboration system, keep old system.

The question at hand was if the quality of the system was good enough to keep it, or if it should be modified or replaced by an ERP-based solution. In order to make a well informed decision, it was decided to analyse the quality of the system by using the extended influence diagram based upon the ISO/IEC 9126 standard.

All subcharacteristics were analysed in order to ensure a correct decision. However, this paper only describes the evaluation of the Security and Interoperability subcharacteristics according to section VI. The evaluation of the remaining subcharacteristics was performed analogously.

B. Scenario 1: Keep old system

This first scenario requires no changes, but rather states that the system should be kept the way it is. It is depicted in Figure 19. The windows-based solution is the result of several years of continuous development by a small, highly experienced staff. Inhouse personnel is also responsible for the system's operation and maintenance. There are two main system components, denoted "communication" and "collaborative management", and they both offer data storage and business functionality. Two separate graphical user interfaces, one for teachers and one for students, are provided.

C. Scenario 2: Modify system

The second scenario entails a modification of the existing collaboration system, cf. Figure 20. The scenario includes replacing the old system components with a more modern three-tier architecture featuring separate components for Graphical User Interface (GUI), Data, and Functions. This greatly promotes maintainability and increases the overall quality of the architecture making it more interoperable.

Additionally, a mobile user interface is created. This forces the collaboration system to interoperate with an external telecommunication system to be able to send text messages regarding for instance rescheduling of lectures or notification on assignment grades. Another new feature is that all traffic is encrypted, the platform introduces an encryption service. Also the operating system changed from Windows to Linux. Apart from the abovementioned, this scenario adds no new functions to the existing solution presented in the as-is scenario. All development, operation and management of the collaboration system is still performed inhouse by the same staff.

D. Scenario 3: Replace system

In the third scenario, the collaboration functionality is implemented in an extension of the Enterprise Resource Planning (ERP) system currently used by the university, cf. Figure 21. The ERP vendor offers virtually the same functions as the first scenario, but execution runs on a windows-based platform with some more security functions, specifically through the use of a firewall. An added benefit with this scenario is the ability to automatically exchange user data between the collaboration system and the human resources department, leading to less duplicate data and thereby a higher overall data quality. A major difference from the other two scenarios lies in the supporting IT organization. The maintenance process uses the same inhouse personnel as in the other scenarios, but the system development was carried out by staff from the ERP vendor. This is a mixed blessing. On the one hand the development team of the ERP-vendor is highly skilled and follows a meticulously documented development process producing structured and high quality source code. This benefits code maintenance as well as system interoperability. On the other hand, the separation of the

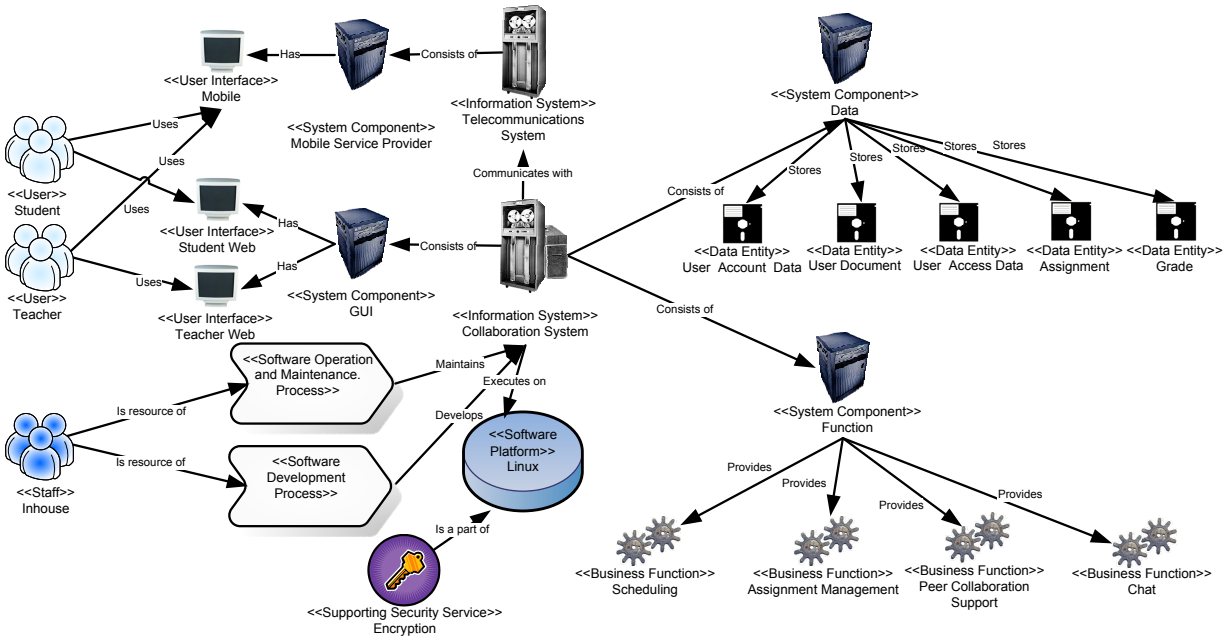

Figure 20. Scenario 2 for the collaboration system, modify system.

development and the maintenance staff means that the maintenance staff has less first-hand knowledge about the inner workings of the system, making it harder for them to localize bugs and correct system errors.

E. The analysis using ISO/IEC 9126

In this subsection, the ISO 9126 based extended influence diagram from section VI is applied on the respective change scenarios from the previous subsection. The result of this will show which scenario has the highest system quality.

All nodes connected to the decision node in the diagram are defined and directly controllable for the decision maker, thus the next step is to collect the values for these nodes.

The values for all the directly controllable nodes are collected as described in ISO/IEC 9126 part 2 and 3. For instance, the metric *access auditability* is measured by specifying the number of logged access types as compared to the logging requirements and the node *data exchangeability (data format based)* is measured as the percentage of correct interface data formats exchanged with other systems.

As presented in section VI there is a conditional probability distribution associated with each node in the extended influence diagram. For the directly controllable nodes these distributions contains the values of the collected data. In Figure 22 and Figure 23 the conditional probability distributions for the directly controllable nodes affecting security and interoperability in the three scenarios are presented.

In this example case, as in many others, the collected measurements are not completely certain, e.g. during the data collection process there is an uncertainty when specifying the number of logged access types for the

Scenario Selection		Scenario 1	Scenario 2	Scenario 3
Access Auditability	High	0,1	0,1	0,8
	Medium	0,1	0,8	0,1
	Low	0,8	0,1	0,1

Scenario Selection		Scenario 1	Scenario 2	Scenario 3
Access Controllability	High	0,1	0,1	0,8
	Medium	0,1	0,1	0,1
	Low	0,8	0,8	0,1

Scenario Selection		Scenario 1	Scenario 2	Scenario 3
Data Corruption Prevention	High	0,1	0,1	0,8
	Medium	0,1	0,8	0,1
	Low	0,8	0,1	0,1

Scenario Selection		Scenario 1	Scenario 2	Scenario 3
Data Encryption	High	0,1	0,8	0,8
	Medium	0,1	0,1	0,1
	Low	0,8	0,1	0,1

Figure 22. Conditional probability distributions of the directly controllable chance nodes affecting security in the three scenarios.

access auditability. As is shown in Figure 22, the choice of scenario 2 (i.e. implementing new components for GUI, Data, and Function) will probably lead to a high level of access auditability. This means that there is some uncertainty regarding what the outcome will be, in this case, 20% uncertainty is divided between the medium and low states. The same reasoning goes for the other variables and the other scenarios.

As presented in section VI and in Figure 24, all the indirectly controllable nodes in the extended influence diagram have conditional probability distributions based on an unweighted average function.

Given the extended influence diagram and the conditional probability distributions, the expected value and standard deviation of the system quality node is calculated for all three decision alternatives. There are several tools available for such calculations, e.g. Hugin [18] and GeNIe

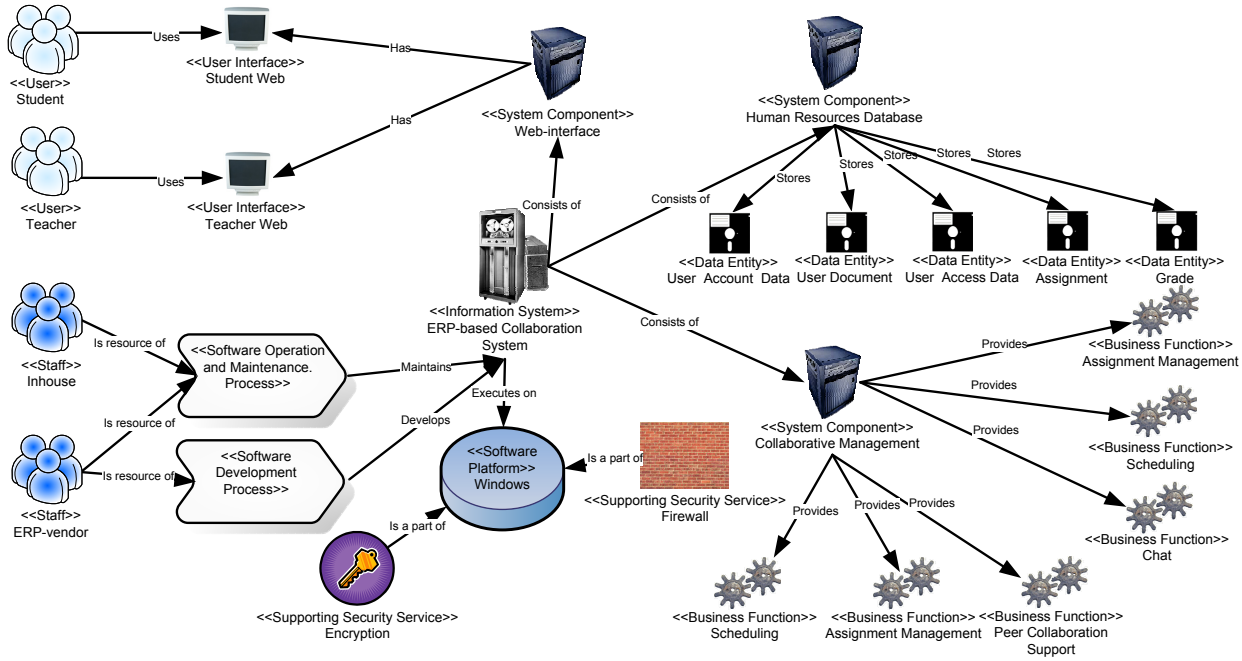

Figure 21. Scenario 3 for the collaboration system, replace system.

Data Exchangeability (Data format based)		High									…			Low
Data Exchangeability (User's success attempt based)		High			Medium			Low			…			Low
Interface Consistency (Protocol)		High	Medium	Low	High	Medium	Low	High	Medium	Low	High	Medium	…	Low
Interoperability	High	1	1	0	1	0	0	0	0	0	1	0	…	0
	Medium	0	0	1	0	1	1	1	1	1	0	1	…	0
	Low	0	0	0	0	0	0	0	0	0	0	0	…	1

Figure 24. The conditional probability distribution for the node interoperability is based on the unweighted average function.

Scenario Selection		Scenario 1	Scenario 2	Scenario 3
Data Exchangeability (Data format based)	High	0,1	0,1	0,8
	Medium	0,8	0,8	0,1
	Low	0,1	0,1	0,1

Scenario Selection		Scenario 1	Scenario 2	Scenario 3
Data Exchangeability (User's success attempt based)	High	0,1	0,1	0,8
	Medium	0,1	0,8	0,1
	Low	0,8	0,1	0,1

Scenario Selection		Scenario 1	Scenario 2	Scenario 3
Interface Consistency (Protocol)	High	0,1	0,8	0,1
	Medium	0,8	0,1	0,8
	Low	0,1	0,1	0,1

Figure 23. Conditional probability distributions of the directly controllable chance nodes affecting interoperability in the three scenarios.

[20]. In this example the GeNIe software developed by the Decision Systems Laboratory at University of Pittsburgh was used. The influence diagrams can be represented graphically, and conditional probability distributions can be assigned to the nodes. The software also helps calculating and presenting the results.

The results from the calculations are presented in Figure 25 and Figure 26. The GeNIe screenshot, Figure 25, shows the values of all security and interoperability nodes for Scenario 1. The diagram, Figure 26, shows the aggregated level of system quality for all three scenarios, as well as the uncertainty associated with the analysis.

The analysis results provides support for the decision maker in the choice between the two alternatives: a) choose a scenario based on the analysis result or b) increase the certainty of result in order to provide better decision support in the selection between the scenarios.

In order to increase the precision of the analysis, it would be necessary to put more effort into data collection.

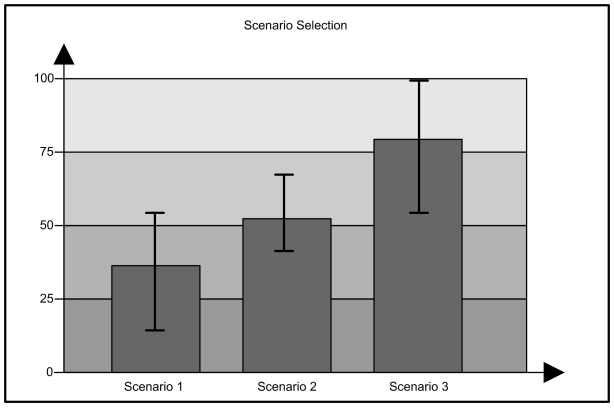

Figure 26. The grey bars show the utility expressed in terms of level of system quality. The narrow black bars indicate degree of uncertainty of the result.

VIII. Conclusions

This article has proposed the use of extended influence diagrams to support the analysis of system quality. The syntax and semantics of extended influence diagrams were

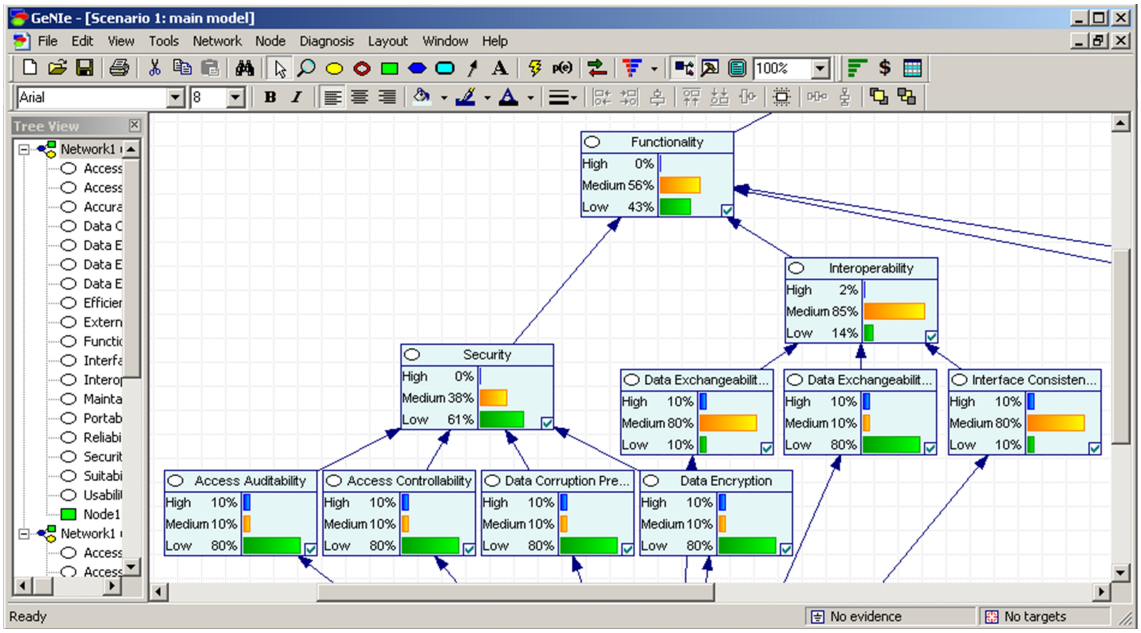

Figure 25. GeNIe screen shot of the results for the security and interoperability part of the system quality extended influence diagram for scenario 1.

presented and a method for their construction and use was described. An extended influence diagram based on the ISO/IEC 9126 was presented. To demonstrate the applicability of extended influence diagrams for software quality analysis a ficticious example demonstrated how an IT decision maker faced with an investment decision used extended influence diagrams for decision analysis.

REFERENCES

[1] R. A. Howard, "Decision analysis: Practice and promise," *Management Science*, vol. 34, no. 6, 1988.

[2] P. Johnson, R. Lagerstrom, P. Narman, and M. Simonsson, "Extended influence diagrams for enterprise architecture analysis," in *Proceedings of the Tenth IEEE International EDOC Conference (EDOC)*, 2006.

[3] ——, "Enterprise architecture analysis with extended influence diagrams," *Information System Frontiers*, 2007.

[4] ——, "System quality analysis with extended influence diagrams," *Proceedings of the 11th IEEE Conference on Software Maintenance and Reenginering (CSMR) - Special Session on System Quality and Maintainability (SQM)*, 2007.

[5] R. Shachter, "Evaluating influence diagrams," *Operations Research*, vol. 34, no. 6, 1986.

[6] ——, "Probabilistic inference and influence diagrams," *Operations Research*, vol. 36, no. 4, 1988.

[7] R. A. Howard and J. E. Matheson, "Influence diagrams," *Decision Analysis*, vol. 2, no. 3, 1983.

[8] F. V. Jensen, *Bayesian Networks and Decision Graphs*. Springer, 2001.

[9] R. Neapolitan, *Learning Bayesian Networks*. Pearson Education, 2004.

[10] *ISO/IEC TR 9126-1 International Standard - Software Engineering - Product Quality - Part 1: Quality model*, International Organization for Standardization, 2001.

[11] *ISO/IEC TR 9126-1 Technical Report - Software Engineering - Product Quality - Part 1: Quality model*, International Organization for Standardization, 2001.

[12] *ISO/IEC TR 9126-2 Technical Report - Software Engineering - Product Quality - Part 2: External Metrics*, International Organization for Standardization, 2003.

[13] *ISO/IEC TR 9126-4 Technical Report - Software Engineering - Product Quality - Part 4: Quality in use metrics*, International Organization for Standardization, 2004.

[14] A. Rubenstein, *Modeling Bounded Rationality*. The MIT Press, 1998.

[15] P. Liu, P. Ammann, and S. Jajodia, "Rewriting histories: Recovering from malicious transactions," *Distributed and Paralell Databases*, vol. 8, 2000.

[16] S. Poslad and M. Calisti, "Towards improved trust and security in fipa agent platforms," in *Autonomous Agents 2000 Workshop on Deception, Fraud and Trust in Agent Societies*, 2000.

[17] M. Scriven, "Definitions in analytical philosophy," *Philosophical Studies*, vol. 5, no. 3, 1954.

[18] *HUGIN API Reference Manual, Version 6.3*, Hugin Expert A/S, 2004.

[19] R. Lagerstrom, P. Johnson, and P. Narman, "Extended influence diagram generation for interoperability analysis," *Proceedings of the Interoperability for Enterprise Software and Applications Conference (I-ESA)*, 2007.

[20] D. S. Laboratory, *Genie On-Line Help*, 2006.

[21] N. Fenton and S. L. Pfleeger, *Software metrics: a rigorous and practical approach*. PWS Publishing Co., 1997.

[22] R. A. Khan, K. Mustafa, and S. I. Ahson, *Software Quality: Concepts and Practices*. Alpha Science, 2006.

[23] J. A. McCall, P. K. Richards, and G. F. Walters, *Factors in Software Quality*. General Electric Co., 1977.

[24] B. Boehm, *Characteristics of Software Quality*. American Elsevier, 1978.

[25] D. C. RB Grady, *Software metrics: establishing a company-wide program*. Prentice-Hall, 1987.

Pontus Johnson is Associate Professor at the Department of Industrial Information and Control Systems at the Royal Institute of Technology (KTH) in Stockholm, Sweden. He is the research leader of a group of fifteen researchers and PhD students

focusing particularly on the analysis of enterprise architectures. In the frame of the department, Pontus also supervises a number of PhD students. He is the author of a number of journal articles, international conference articles and books. He is also active as a consultant in the area of enterprise architecture. Pontus received his MSc from the Lund Institute of Technology in 1997 and his PhD from the Royal Institute of Technology in 2002.

Robert Lagerström holds a MSc in Computer Science from the Royal Institute of Technology (KTH). Robert is currently a PhD Student at the Department of Industrial Information and Control Systems at KTH in Stockholm, Sweden. His research focus is on enterprise architectures (EA) and decision analysis of maintainability using EA-models. Robert is also a member of the Swedish Chapter committee of INCOSE (International Council on Systems Engineering).

Per Närman holds a MSc E.E. from the Royal Institute of Technology (KTH). He is currently a PhD Student at the Department of Industrial Information and Control Systems at KTH in Stockholm, Sweden. Per conducts research within the field of enterprise architecture analysis and his research will result in a system quality analysis framework, an enterprise architecture metamodel connected with said analysis framework and a methodology that supports resource-efficient modeling.

Marten Simonsson holds a MSc E.E. from the Royal Institute of Technology (KTH). He is currently a PhD Student at the Department of Industrial Information and Control Systems at KTH in Stockholm, Sweden. He focuses his research on the area of IT governance with the overall goal to improve the COBIT framework for maturity assessments of IT organizations. Mrten is also active as a consultant in the area of enterprise architecture and IT governance.