We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

7

Quality Model – Master Plan and DNA of an Information System

Finne Auvo
University of Jyväskylä, Finland

1. Introduction

The goal of the chapter is to give a refined definition for the quality of information system as a technical artifact and based on that statement present a complete conceptual framework for quality modeling. Further, the chapter shows how a quality model as a master plan for information systems can drive and control the entire development process.

1.1 Information system and its context - Models and objects of modeling

Every theory has its surroundings and postulates. So has a theory about quality models, and it is better to make the main lines of these ideas explicit before presenting the theory itself. A human made information system (IS) as a technical artifact exists and operates always in the context of societies, organizations, personal lives etc. It is a tool used for gathering, storing, processing, presenting and exchanging (communication) information. These activities can be termed "information behavior" (Allen et al., 2011). Accordingly the context of an information system has a two-tiered structure (Figure 1). The inner tier, information behavior, is subordinate to the outer tier, human society. Information in general is used to support human activities, and technical information tools, in turn, are used to enhance the use of information.

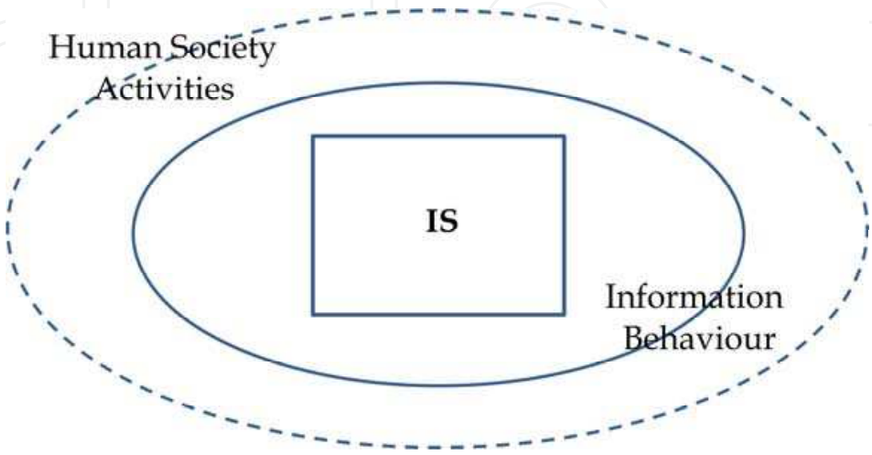


Fig. 1. Two-tiered context of information system

In this context both information and information system are supposed to have meaningful functions, and because of these purposes human actors have various wants, needs and expectations about the information itself and the supporting system's behavior. This view is implicit for example in the writings where sense-making theory is applied to information system design (e.g. Muhren et al., 2008). Sense-making provides means to understand how information in general is used by humans (see e.g. Savolainen, 2006), which in turn should be reflected in information tools design. Another theory frequently used (e.g. Silva, 2007 and Macome, 2008) to explain the interplay between information system and its context is Actor Network Theory (ATN). ATN views technology as part of a network of human actors and nonhuman artifacts. And still one interesting framework has been developed by Alter (1999, 2008). He views information systems in connection with work systems that the former serve, and in the end information systems as special cases of work systems. Human society is full of work systems "in which human participants and/or machines perform work (processes and activities) using information, technology, and other resources to produce specific products and/or services for specific internal or external customers" (Alter 2008, 451). The definition of work system comprises projects, supply chains, web sites, etc.

In everyday life and scientific literature actors' expectations for information and information systems are commonly called requirements, and a part of them more precisely quality requirements. Each IS has a life course that contains various cyclical or repeating processes, like system development, moving from one platform to another, etc. ISO/IEC 12207 (2008), for example, gives a good picture of software lifecycle processes. In the course of these processes there are several points where requirements are captured and goals set, something – usually a system or component - to fulfill the requirements modeled, implemented and put into operation, and finally the results measured and evaluated. The words "modeling" and "designing" are in this chapter used interchangeably. And a "model" means the product of modeling or designing process, an abstraction of or a blueprint for something to be realized. The difference between requirements and a model based on them is that requirements express needs and desires and are often less structured and consequently written in the form "x is needed" or "x must be y", whereas a model is structured and statements are in indicative form like "x is y".

Traditionally modeling in software engineering has centered on the system itself as a product of development, the development process or the entire system life course (Figure 2). With respect to these three alternatives this chapter focuses on information system as a technical artifact seen in its context of development and use, primarily human activities. It does not go into the area of system development process or IS life cycle models. Consequently, the quality models that are discussed can be categorized as product quality models.

Modeling can take place on different levels of abstraction. And the actual object (X) that is modeled can be any real thing, not only an entity or process, but even a state of affairs as this chapter will show. After modeling and implementing several Xs one can create a general model of Xs or of certain type of Xs that consequently constitute theories of X. Finally, instance and general level models can be used to create a meta-model, a model for X-models that can be regarded as an even higher level of theory. The other way around, higher level model can be used as template for creating lower level models. By iterating this

kind of model generation, from bottom up and from top down, models on all levels grow better and better by the time and experience. Figure 3 depicts this relationship between levels of modeling, and at the same time between practice and theory.

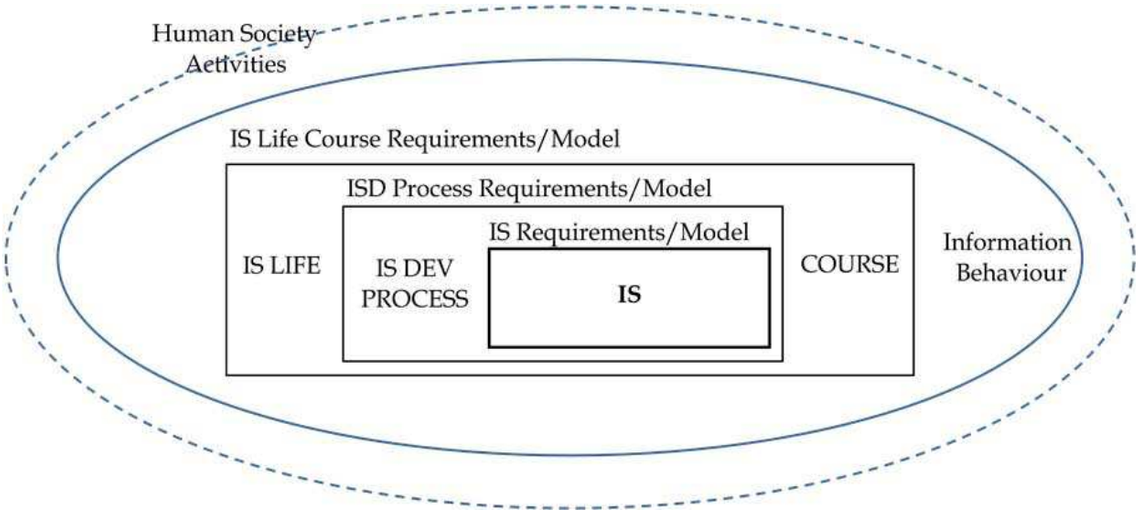


Fig. 2. Traditional objects of modeling in software engineering

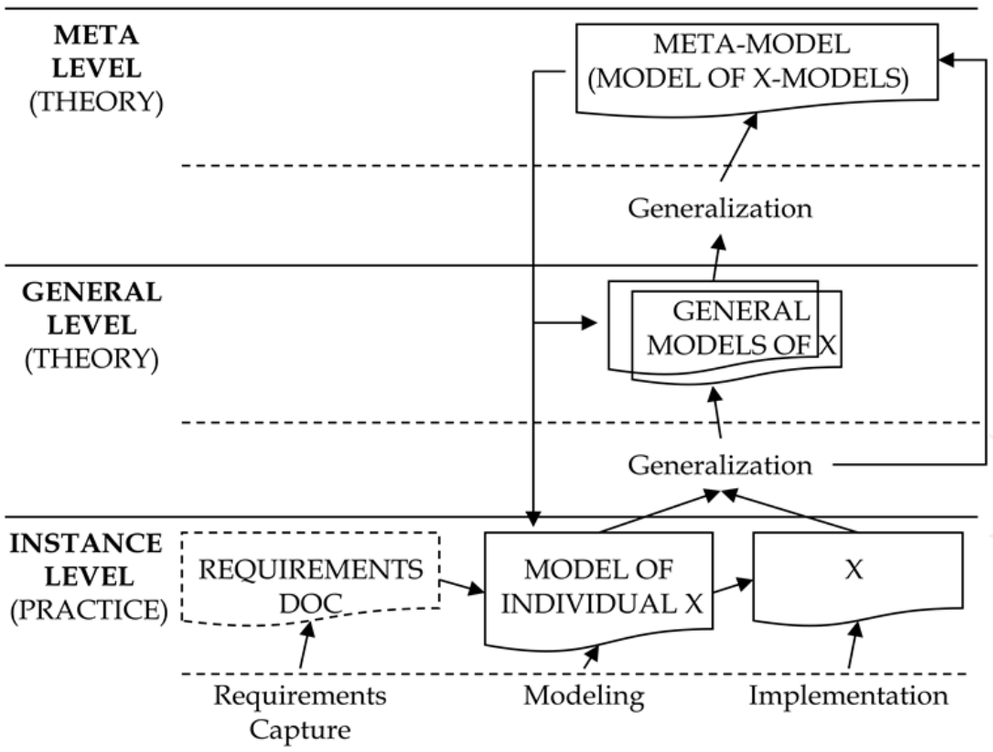


Fig. 3. Levels of modeling, practice and theory

This chapter is about quality requirements, quality models and their DNA like role in system development. In order to be complete and avoid misunderstandings, an account of

quality requirements, quality models and their DNA like role in system development has to start from the meta-level.

2. The essence of quality - Requirements and quality requirements

Reeves and Bednar (1994) have discussed different definitions of quality in business context. They state that the concept of quality has had multiple and often unclear definitions and look more closely at four of them: quality as 'excellence', 'value', 'conformance to specifications', and 'meeting expectations'. Excellence means meeting the highest criteria in some area like intelligence, strength etc. The value aspect introduced price, or value, as an additional determinant of consumer's decision. Different compatibility requirements in production of component based machines lead to equaling quality with conformance to specifications and to making quality measurable. Finally the most pervasive definition 'meeting customer's expectations', according to Reeves and Bednar (1994), grew out of services marketing. It is also the most complex definition and most difficult to measure. Reeves and Bednar (1994) conclude that a global definition of quality does not exist and different definitions are appropriate in different contexts. The IEEE Standard Glossary of Software Terminology offers a two part definition of system quality: "the degree to which a system, component or process meets specified requirements" and "the degree to which a system, component or process meets customer or user needs or expectations. It coincides with the fourth definition discussed by Reeves and Bednar (1994).

Defining quality can be started from the viewpoint that in very general terms information system quality can be seen as determined by the existence and intensity of something pertaining to the system, identifiable and desired by actors and stakeholders. This point of view is in a way similar to the definition of quality as 'meeting expectations' above. What is desired is referenced to by using adjectives and abstract nouns and commonly interpreted as characteristics or features that reside inside and constitute an integral part of the entity (system) being described. Quality definitions like in ISO 9126 reflect this viewpoint. It gives in the annex a general definition for quality taken from ISO 8402:1994 (replaced now by ISO 9000:2000): "the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs" (ISO 9126 (2001, 20)). Analysis of the meaning of adjectives and abstract nouns used for qualities discloses, however, that they refer actually to certain states of affairs or relationships between the system and things in its context. By taking, for example, ISO 9126 quality model's six main characteristics functionality, reliability, usability, efficiency, maintainability and portability as examples, one can see that they all refer to a relationship between information system and its context. Functionality is the capability of software to provide functions which meet stated and implied needs (ISO 9126, 7). It is clearly a relationship between the product, its users and business processes they have to carry out. Reliability is defined as the capability of software to maintain a specified level of performance under specified conditions (ISO 9126, 8). Again it is about a relationship, this time a relationship between the product, a specific instance of using it and certain conditions. Usability is the capability of software to be understood, learned, used and attractive to the user when used under specified conditions (ISO 9126, 9). This characteristic relates the product to the user and certain conditions. Efficiency is the capability of software to provide appropriate performance, relative to resources used and under stated conditions

(ISO 9126, 10). It relates the product to resources and use under certain conditions. Maintainability is the capability of product to be modified and adapt to changes in environment, requirements and functional specifications. Again it is clearly about the product in relation to its context. Finally portability is defined as the capability of software to be transferred from one environment to another (ISO 9126, 11).

The search for the essence of quality can as well be started from inside the system. Taking an internal view, any information system can be described exactly and without remnants by indicating its architectural type, programming language used, design patterns, layers and packages, by listing procedures and methods, records in the database, series of instructions, and so on. This kind of description does not, however, explain WHY these constituents are required. In some cases an element is needed to create another element. But for what is the latter needed? Following the chains of *WHY*s one comes in the end out of the system internals into some desired relationship between system and its context, even if it is just a relationship between system and individual actor. Accordingly, for to explain *WHY* a design pattern, method, etc. is required or needed, these relationships must be described and understood. After finding the raison d'être of internal constituents in system-context relationships, the constituents themselves can be viewed as contributing factors in the former.

The fulfilment of quality requirements is not only dependent of internal system characteristics. Studies on information system quality show that external things can also have an impact on the desired relationships. Narasimhaiah and Lin (2010), for example, have studied external contributors. They focus on organizational and individual human factors associated with quality attributes like reliability, ease-of-use, maintainability, usefulness and relevance. They found as external determinants of software quality things like attitude of management, responsiveness and capability of IS department and capability of users themselves. Each of the determinants was further decomposed into smaller items. Capability of users, for example, consisted of users' knowledge in the system, training received, involvement in or resistance to the system, and technical competency. When it comes to super-attributes like sustainability the importance of external contributors is evident.

Figure 4 depicts the above viewpoints. It lists along the upper part of the ellipse a number of terms used in literature for referring to those subcategories of requirements that are commonly regarded as quality requirements. Functional requirements are added to the set on basis of the discussion above. Each individual requirement (NR1, G1 and FR1) or "desire" points to a state of affairs or relationship (R1, R2 and R3) between information system and its context. At the same time these requirements point to certain concrete system features (e.g. architecture and method) and things (T1, T2 and T3) outside the system. The former, desired relationships, explain the need for the latter, concrete system features and external conditions. Accordingly, if a developer starts asking in respect of any internal system feature that is under design or implementation, why it is required, he or she finally always traces back to some desired relationship between the system and its environment. This subordinate status of system internals and externals compared to the expected system-context relationships explains the meaning of the expressions "in the first place" and "in the second place" in the following definition paragraph. The two-tiered nature of system context, discussed in section1, is represented in Figure 4 by the two circles around the system rectangle.

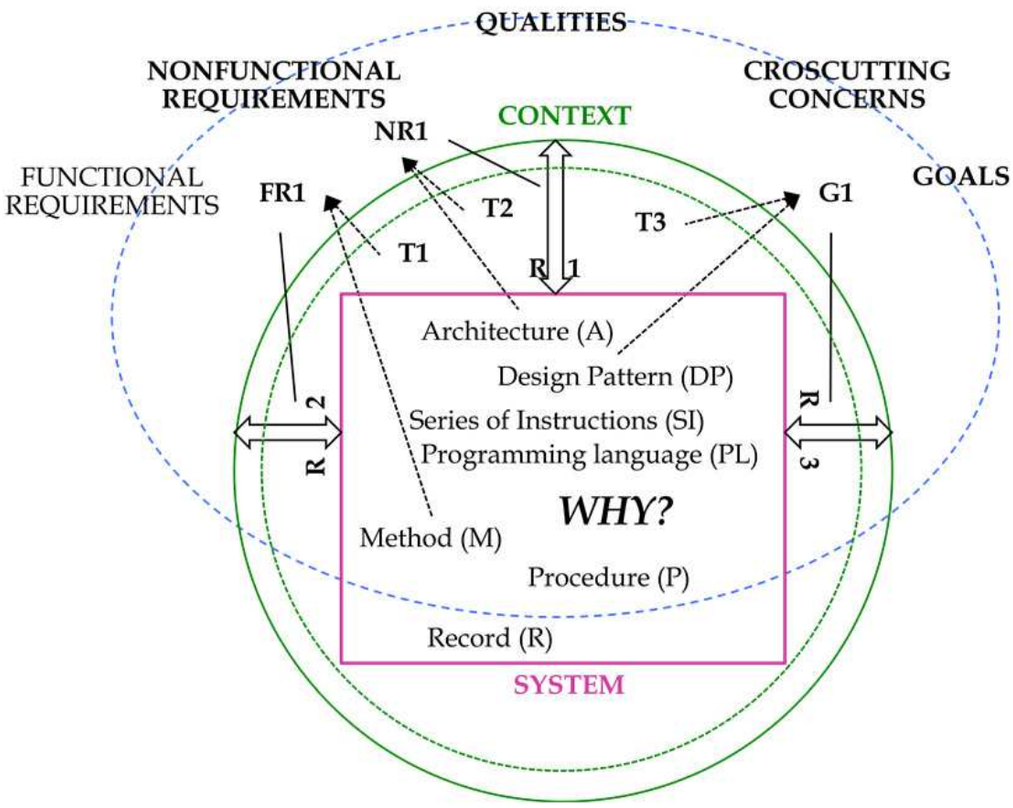


Fig. 4. Quality requirements and contributors

Quality of information system as a technical artefact in its context of development and use is in the first place determined by the existence, lack, intensity or number of desired relationships between information system or system constituent and its context. It can also be characterized as a desired state of affairs. In the second place quality of information system is determined by the existence, lack, number of or intensity of elements, inside or outside the system, contributing to the realization of the desired relationships. These elements get in a way there "justification" or "raison d'être" through the desired system-context relationships. Consequently individual qualities as well as overall quality of information system must be defined and measured in terms of these relationships.

'Requirement' as such is a broader notion meaning anything required, be it a certain quality, or something needed to realize it, or something else needed in the system for some reason. The main difference between quality attributes and other attributes, or quality requirements and other requirements, is that the former refer in first place to desired relationships between information system and its context and have more importance (priority) to actors than the latter, and the latter can often be derived from the former. Consequently quality requirements constitute the core of requirements for an information system. Priority, definition and measured level of implemented quality requirements are often relative to use case, actor, or some feature of the context. Therefore actors usually agree on goals regarding to what degree quality requirements need to be met.

3. A comprehensive quality model

Given the definition of information system quality put forward in previous section it is obvious that the design for a quality technical artifact, being created and functioning in its context of development and use, has to cover quite a number of different aspects. Describing a state of affairs or relationship takes much more than describing an entity and its attributes. Case study findings (Finne, 2011) suggest that a comprehensive model, that is needed to fully account for the quality of an information system as an end product of development process, appears to be a hybrid model with six sub-models. It can alternatively be described as an intersection of models. Figure 5 represents a meta-level view of the hybrid model.

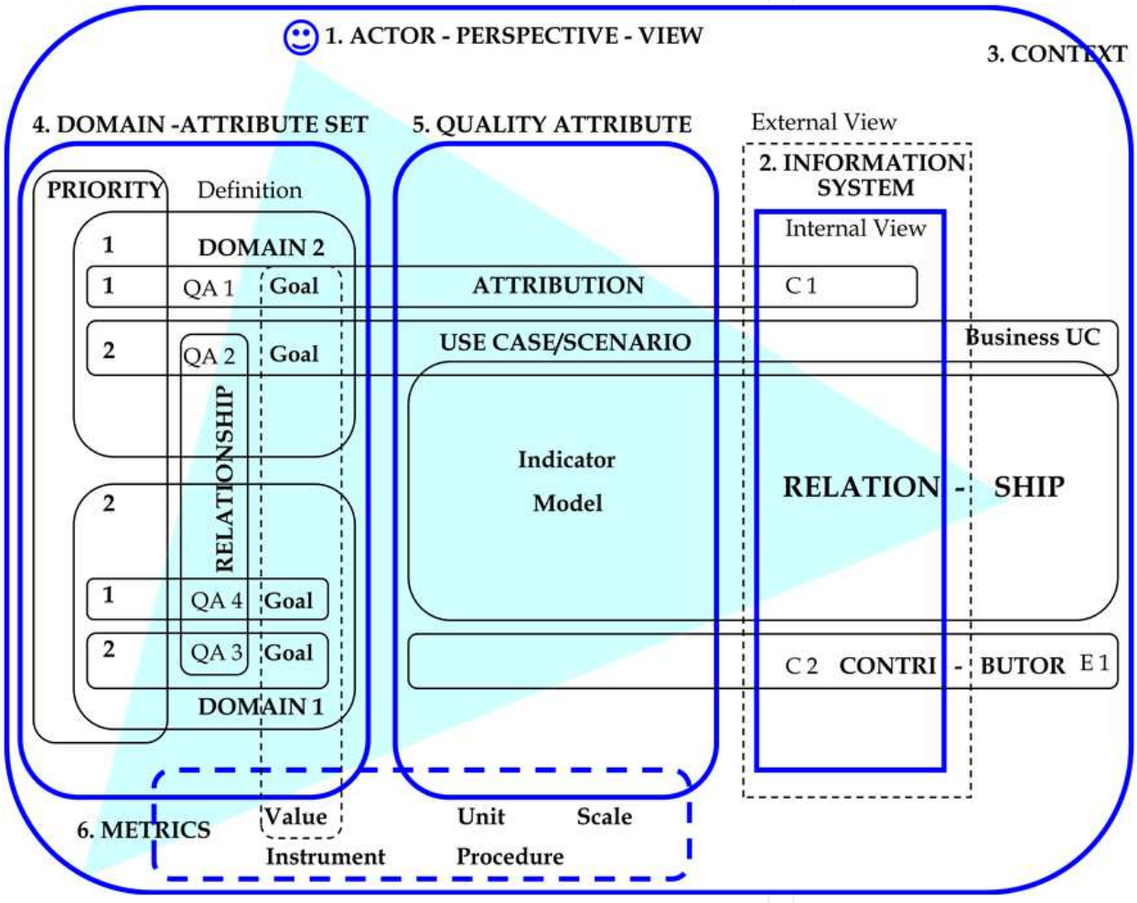


Fig. 5. A meta-level view of information system quality model

The first sub-model, 1) human actors with their perspectives and views (symbolized by the filled triangle without borders) is actually part of an activity or process (quality modeling or system development) model. Actors are typically seen as elements of activity models. Next, to gain an understanding of the target information system on necessary levels of abstraction 2) an information system model is needed. To deal correctly with 3) context one needs to model it to some extent. Next, the 4) domain and quality attribute set reflects the core requirements and areas of concern for the stakeholders, and can be viewed as a model of overall system quality. Individual 5) quality attribute models are in essence models of

desired system-context relationships or states of affairs. Finally, the 6) metrics part is a model by itself. It describes procedures and means of determining if the intermediate or end product of development process complies with the design in respect of quality. Following paragraphs give a detailed definition to meta-model elements. D1 and D2 stand for two sample domains. QA1, QA2, QA3 and QA4 are individual quality attributes. C1 and C2 symbolize system constituents, and E1 symbolizes an entity in the context. UC in "Business UC" stands for use case. Goal means a goal value set for a quality attribute. The overlapping of rectangles symbolizes:

- PRIORITY: Both domains and attributes are prioritized.
- ATTRIBUTION: A quality attribute (QA1 in the figure) in the attribute set can be attributed to the information system as a whole or to some of its constituents (C1 in the figure). The attribution of qualities starts when the domain-attribute set is created and is reviewed during modeling of individual attributes.
- USE CASE/SCENARIO: Use cases comprise system and business use cases, the latter being part of the context. An attribute (QA2 in the figure) can be initially connected to use cases already during creation of domain-attribute set. Attributes can have different order of priority in connection with different use cases.
- RELATIONSHIP (on the left): Quality attributes are interrelated in many ways.
- RELATIONSHIP (on the right): The majority of quality attributes refer in first place to desired relationships between information system and its context. These relationships become visible and are defined in connection with use cases and scenarios.
- CONTRIBUTOR: Internal (C2 in the figure) and external (E1 in the figure) contributors determine for their part to what extent the desired relationships are met. Contributors are system constituents or things in the system context.
- VALUE/GOALS: Goals are target attribute values.
- METRICS: Metrics can be designed for individual quality attributes or overall quality (taking into account the whole attribute set).

People or groups of people who have some meaningful relationship with the information system under scrutiny are called *actors*. They are affected by the qualities of the information system or its products. A general naming for actors that can also be used is "stakeholders". Some actors may never use the system, but are anyway somehow interested in it or affected by it and its products. The term "*informants*", in turn, means a sub-set of actors who actually participate in quality modeling or quality measuring and give some relevant information. Actors are part of the information system context.

Human actors have certain perspectives on and views of information system and its context that are reflected in resulting quality models. Each quality model element can in fact be traced back to a particular actor or actor group. A *perspective* is characterized by the actor's background, organizational roles and activities, beliefs, values, etc., and actor's relationship to the information system on basis of them. The former are at the same time elements of the information system context. A *view* of information system and its environment, in turn, is characterized by what it excludes and what it includes, i.e. by the constituents or elements visible in the view and their relationships. The elements in a view can be activities and processes as well as other things. A view of system can be concrete (through using or creating the system) or based on system descriptions. It can be general or detailed, partial or complete and so on. A view can be affected by elements constituting the perspective or other

psychological and cognitive factors that determine what an actor wants to see or how an actor interprets what is seen. In terms of quality modeling a particular view usually covers only some parts of the information system and context.

For each actor, the knowledge about context, the knowledge about information systems in general and about the specific system under modeling together with the perspectives given by the roles and work the actor participates, constitute a kind of *frame of reference* that determines the appearance of the system to the actor. Perspectives and views which are presented and shared in the course of quality modeling form the intersection of individual frames. Figure 6 depicts the main elements of actor's frame of reference. The system is depicted on a very high level of abstraction as a structure built from different constituents, having contents and acting in a particular way. T1, etc. stand for "things" and S1, etc. for "systems" in the context of system and actor.

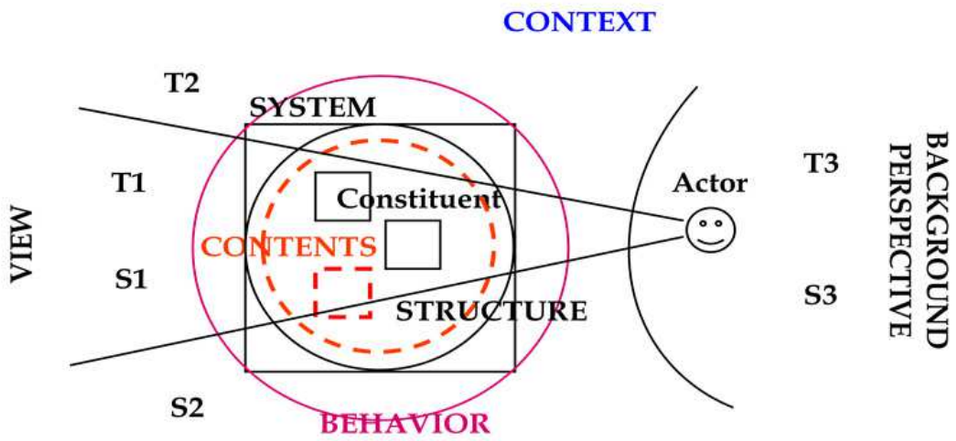


Fig. 6. Actor's frame of reference

An information system as a technical artifact consists of electronic and non-electronic components (*constituents*), their functioning (*behavior*) and relationships (*structure*). It includes the human and machine interfaces for data input and output. It is used to store, process, produce and present information (*contents*) in order to support human activities, including entertainment. Information products created by the system or serving as input into system are regarded as elements in a wider integrated system or the human information behavior as a whole. From the perspective of product quality modeling the systems elements have no advance justification. The reason or explanation for their existence comes through their ability to contribute for fulfilling quality requirements. Starting from a mere black-box view, gradually, according to the needs of attributing some qualities to lower level components and finding contributors to the qualities from inside information system, more detailed architectural descriptions are used. Good architectural descriptions are worth of gold but unfortunately a rarity in system development projects.

From the perspective of quality modeling *context* comprises all the elements in system environment that are members in the desired relationships between the system and context. These relevant entities can be human or non-human, independent of the spatial or temporal distance. Business model forms an important part of context model.

A *quality attribute* refers in first place to a desired relationship between information system and its context or to a number of connected relationships. The existence and intensity of this relationship determines the level of quality in question. In a domain-attribute set each quality attribute can be given a general *definition* and a *goal value*, and it can be *attributed* or allocated to the system as a whole or some of its constituents. Having goals is essential in defining and assessing quality. Attribution to a constituent means that the particular quality of the constituent determines in fact the same quality of the whole system. Qualities are of high importance to actors and form a prioritized set.

Domain is a field of thought or area of concern to the actors in connection with which a quality attribute or group of attributes is relevant. It groups together related attributes that can be viewed as individual concerns. Each domain in an attribute set or collection is given a general *definition*.

A *domain-attribute set* is a *prioritized* list of all quality attributes ascribed to the information system as a whole or to a particular constituent. The attributes in the set are given a general definition and goal value, grouped according to prioritized domains and related to each other. Attributes themselves are prioritized on the level of the whole set, inside domains or both. Priority is characteristic of quality determining attributes. Different factors, among them actors' perspectives and views, have an impact on the final selection and prioritization of attributes. Quality *domain-attribute collection*, in turn, is a general set or supply of domains and quality attributes that can be used as a source when assembling the system specific domain-attribute set. Specific attribute collections can be created for different types of systems.

Figure 7 shows as an example a domain-attribute set that was used in an EMIS (Education Management Information System) case study (Finne, 2006). Attributes sustainability and suitability are positioned in the middle of the "palette" to underline their composite nature. This kind of presentation helps to identify biases and gaps in system's quality design. In Figure 7, for example, neglected domains are 'architecture and design', 'change' and 'performance'. Predefined, general or system type specific attribute collections, in turn, can act as starting points and checklists ensuring that the experiences of similar information systems and similar environments, or information systems in general, are taken into account. At the same time it must be kept in mind that no listing can cover all possible quality characteristics relevant to all possible information systems. Similarly a fixed and non-controversial categorization of quality attributes is probably impossible.

Business use cases, as part of the business model, represent the processes of an organization with or without explicit reference to supporting information systems. A *system use case*, for its part, represents the use of system per se, without, in the first place, drawing attention to its connection to business processes. The term *scenario*, in turn, refers to particular circumstances or to a flow of events, other than use cases, where the role of human actors as users of an information system sometimes can be non-existent, or not focused on.

Most of the characteristics that are traditionally called 'quality attributes' refer to a desired *relationship* or set of desired relationships *between the information system*, or its constituent, *and* one or more entities in the *context*. This can be regarded as characteristic of quality

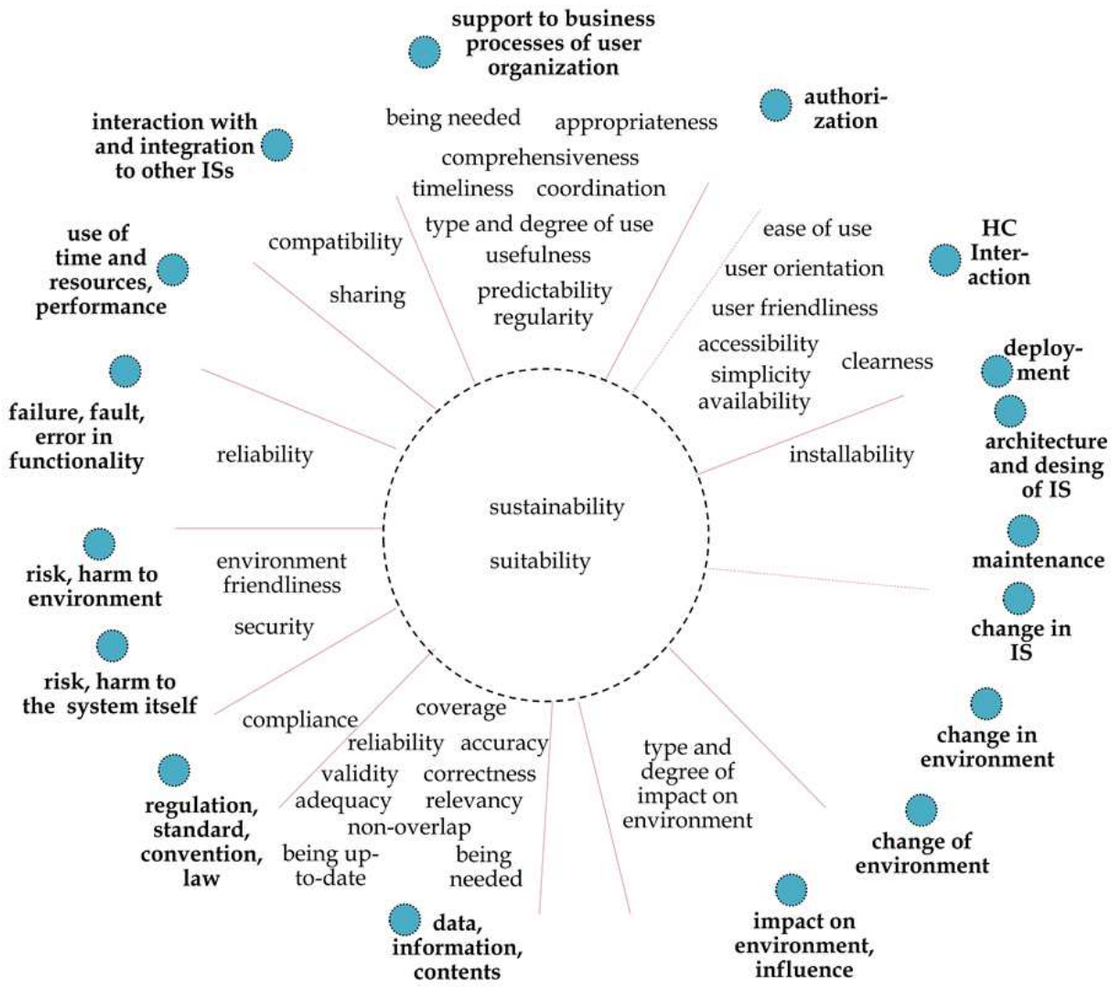


Fig. 7. An example of domain-attribute set

attributes. It can also be named a 'desired state of affairs'. In this way the system is linked with those entities in actors' minds or physically. Certain facts, called *indicators*, indicate the existence and intensity of the relationship in connection with real use cases and scenarios. A quality model can in addition list negative facts that show the lack of the desired relationship. A general formal presentation of the desired relationship is called *model*. It can be given, for example, in the form of an entity relationship (ER) model. Figure 8 gives a simplified example. It is taken from a quality modeling case study conducted in Zanzibar. It is part of the security-attribute-model and shows three entities: a land registration system (LRS), one human actor and one external connected information system. One security related attribute (security mechanism) is attached to LRS and another (ICT skills) to the human actor. When the human actor or external system tries to connect to LRS the latter either denies or allows the connection and shows and hides information according to implemented security rules. This really is a simplified example. In practice one can identify a number of additional relevant attributes. And the description of relationship is more advanced.

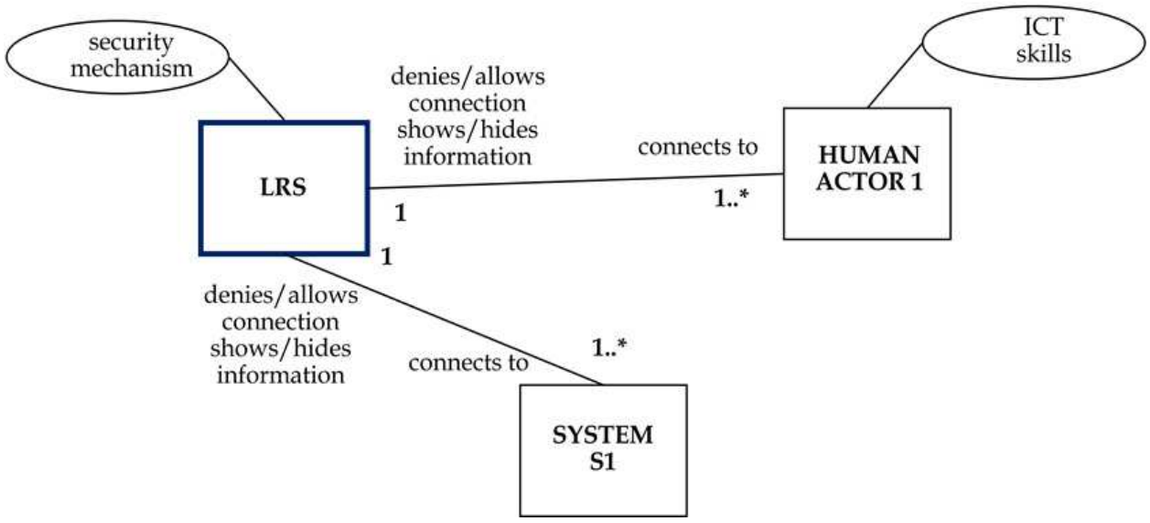


Fig. 8. A simplified ER-model depicting the security system-context relationship

Contributor is a thing *inside or outside* the information system that affects in a positive way a desired relationship between system and its context. It can be alternatively called "factor". The description and understanding of system environment, discussed above, is important in identifying the external contributors. The internal things, in turn, are usually system constituents, behavior (functioning) or structures and consequently part of the system model. Desired relationships are the "raison d'être" of contributors and quality attributes can be said to refer in the second place to the latter. What are the contributors in reality with respect to each quality is in the end a subject of empirical study. Thereafter, based on achieved theory system developers can instantiate the contributor elements in system design.

Attribute relationships are a feature of domain-attribute set. These relationships can be identified by comparing indicators, models, contributors and measured values. Indicator sets, for example, can be separate, overlapping, or one included in another. Contributors, in turn, can be indifferent to one another, cooperating (supporting), or conflicting. In theory there can be as many kinds of attribute relationships as there are relationship types between indicators or contributors. Figure 9 shows a way to depict diagrammatically attribute relationships. It is taken from the same EMIS case study (Finne, 2006) as Figure 6 above. First, the circle at left represents five top ranked quality attributes. The angle of the slices represents the relative priority value (as a number in brackets) of attribute. In the matrix "+" sign stands for a positive contribution. As one can see from the matrix, the attributes are quite independent. Only usefulness is clearly influenced by most of the other attributes. Different signs in the intersection of rows and columns can symbolize different relationships. A similar matrix is used by Khaddaj and Horgan (2005).

A complete arrangement for measuring the existence and degree of a quality, i.e. of a desired relationship between system and its context, includes selection of *instrument, unit, scale, actors, and measurement procedure*. The *object of measurement* is in first place the existence and degree of certain desired system-context relationship represented by the indicators, and in the second place the existence and properties of internal and external contributors. As

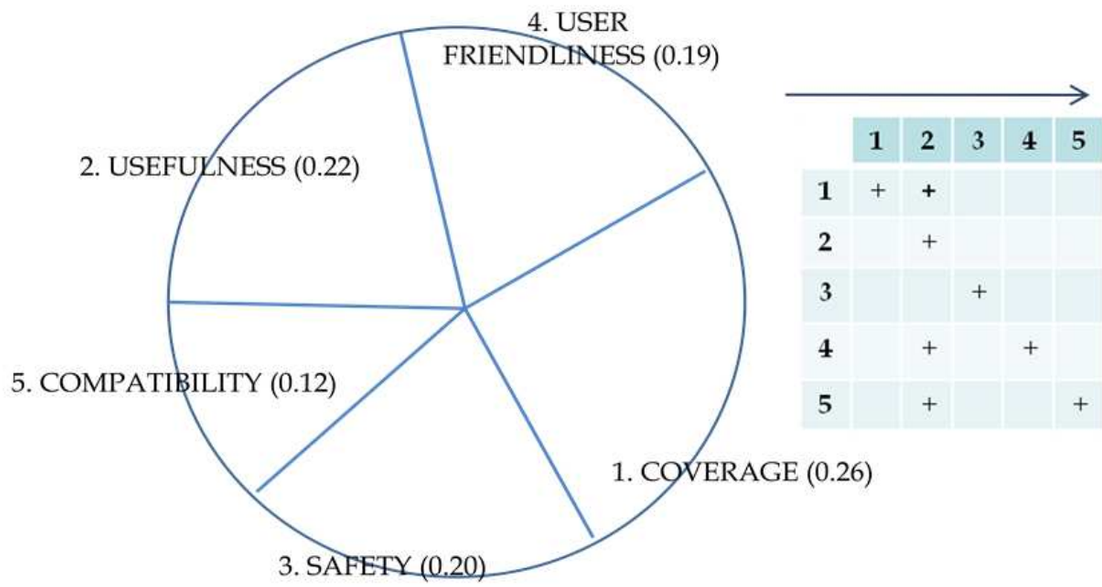


Fig. 9. Example of relationships between top five attributes

was noted above in connection with actor's perspective and view, the objects of measurement are in each case defined, observed and measured in a certain *frame of reference*. The things that cause relativity, i.e. difference of results compared to another frame of reference, constitute together the set of axes of the frame. They include, for example, business and system use cases, business processes, observer's organization and role in it. Relativity is characteristic of information system quality.

There exist many alternative ways of representing graphically the model of a specific quality attribute, or even the entire quality model of a particular system. In general, however, it is often impossible to fit all information into one diagram. This problem has to be solved by representing only core features graphically, by attaching textual descriptions or distributing the information on several diagrams. Figure 10 exemplifies how little information actually fits in a diagram that can be viewed without scrolling. It is again taken from the EMIS study and it summarizes some core features of 'coverage' quality. The attribute refers to the concern of covering with the system all the information that potential users need. It is ranked as number one concern. The general definition has been written by the researcher. Only one indicator is shown in the upper right corner. The model reflects the perspectives and views of researcher, 11 selected informants and the EMIS development team. Attribute belongs to the data-domain and is positively related to usefulness. Of the context elements only an unnamed business process (P1) is visible in the diagram. It represents the large number of business processes where actors use information provided by the system. Coverage has been defined with respect to process data use case. It refers to use of EMIS for processing statistical information for example in order to create different statistical presentations. 100 per cent coverage is set as goal. Measuring instrument, unit and procedure are indicated in the lower left corner of the diagram.

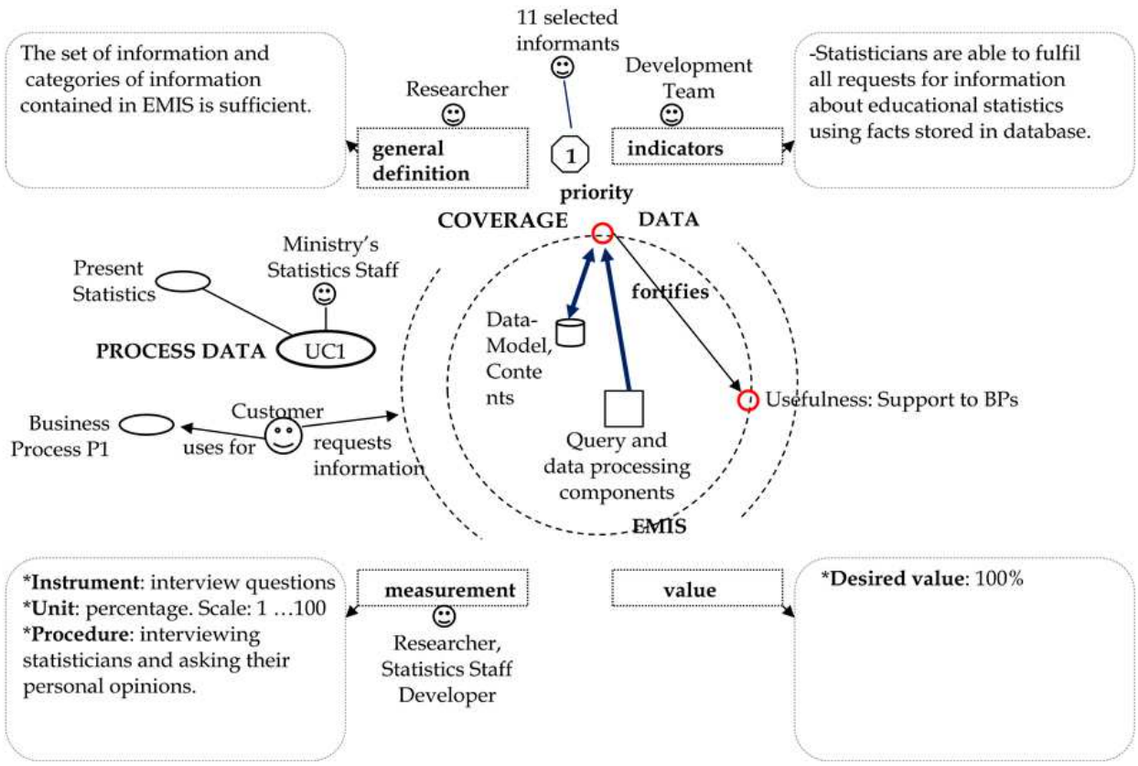


Fig. 10. Graphical representation of coverage attribute model

4. Quality driven development - Quality model as master plan of information systems

Given the view on quality requirements constituting the core of requirements for an information system, an endeavor to fulfill them must also be the core process in system development. During the last ten years the software developer community has seen a rise of different approaches characterized by the word "driven". The experiences in case studies (e.g. Finne, 2011) using the comprehensive model suggest introducing still another approach that could be called simply "quality driven development". It does not only mean that quality design and implementation must be a truly integral part of software development that cannot be given up. It means raising the quality modeling from the role of being just a separate component to the status of an umbrella like driving force. If stakeholders in the end want a quality system, quality is also the issue to start with. All other goals and decisions should be subordinate to it and all other design elements in line with quality design. In this sense quality drives the whole development process, not only for example, architecting. And if quality is understood as realization of a set of desired and most essential system-context relationships, then a quality model can be seen having a DNA-like role and carrying the "genetic information" of system.

In an era that pursues agility it is often feared that the introduction of extra models cause too much overhead to development work in terms of calendar time, money and other resources that are badly needed to address more fundamental challenges. But what can be more fundamental than capturing, prioritizing and realizing core system requirements. Figure 11

shows how the creation and use of quality model elements are positioned in relation to object-oriented software development process (as presented by Jacobson et al., 1999).

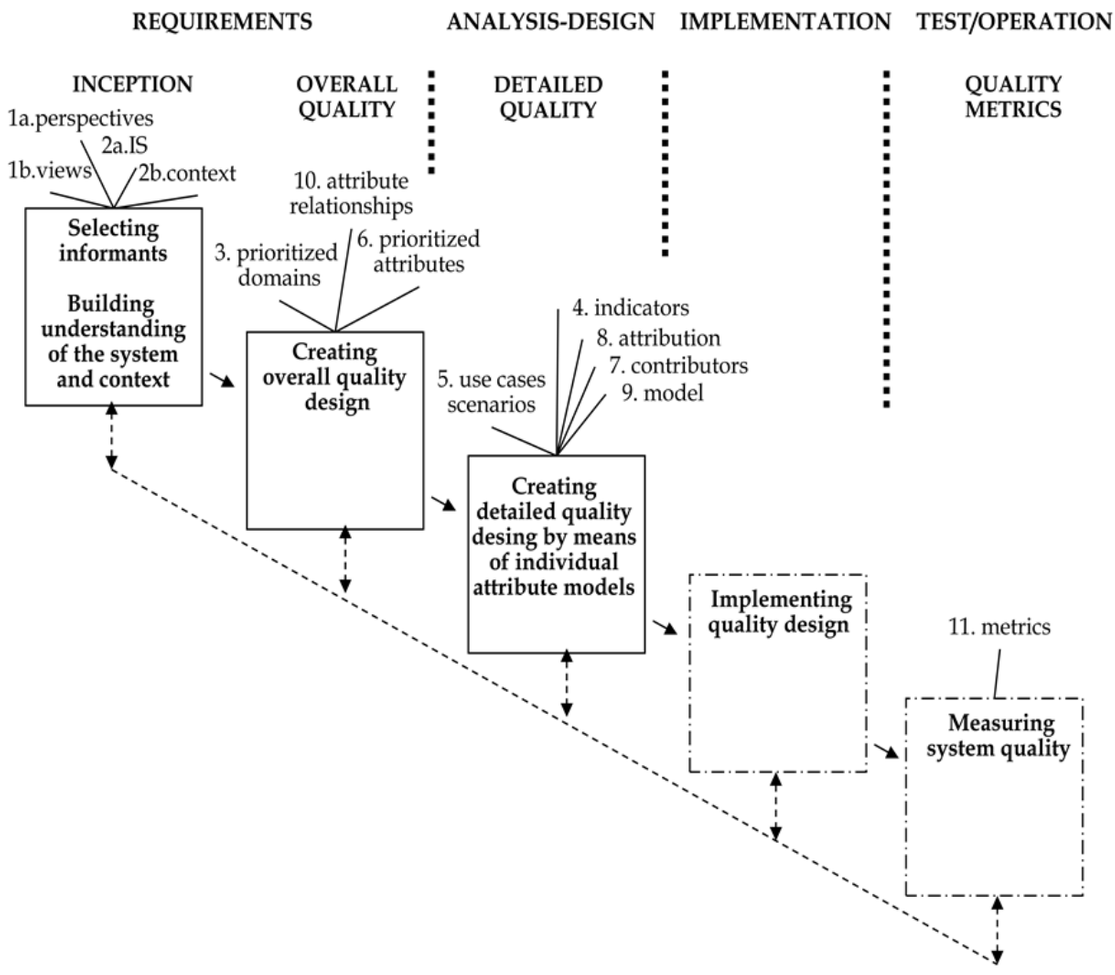


Fig. 11. Quality modeling in relation to object-oriented development process

Awareness of context as well as perspectives and views of actors who participate in system development is mandatory in any kind of development approach. IS, the technical artifact, must always be designed. Further, no system exists separate from requirements that are sometimes conflicting and need to be prioritized. Use cases and scenarios are important stuff even in agile methodologies. Finally every system must be tested, assessed and compared to requirements. In these respects quality modeling does not bring any extra burden to the picture. The point in quality driven development is that quality of an information system is understood to be in first place determined by the existence and intensity of desired system-context relationships and consequently as well defined and measured in terms of these relationships. The quality meta-model, presented in previous section, provides a template for designing the relationships and relating the resulting model to other models needed in system development process. The point is further that a system specific quality model is the arch-model in system development and never left out of sight,

and that everything what is done is done in the name of realizing the quality goals. The following guidelines for applying the quality meta-model are based on experiences of three case studies in East-Africa.

From Figure 11 one can one can see how overall quality design coincides with requirements capture in unified software development model. Jacobson et al. (1999) divide requirements into two categories: functional and non-functional requirements. Functional specifications tell what the system is supposed to do for the users. Non-functional requirements, in turn, correspond to what are traditionally called quality attributes, like performance, availability, etc. The core quality 'usefulness' in fact covers the functional requirements in Jacobson's model. Consequently a well designed and prioritized system specific attribute set with goal values and attribute relationships, can cover the whole range of requirements and act as a guide and driving force for the whole development process. The importance of integration between functional and other requirements is seen for example by Kotonya and Sommerville (1996).

The first task during a phase that can be called "*inception*" is always formation of an actor group that carries out quality modeling. This group is so important that it is positioned as the first element in quality meta-model followed by the initial understanding of the system and its context. The latter things have an impact on the selection of first group members. Even the first understanding of the system is inevitably an interpretation made by some human actor(s). If the group is formed when the project is initiated and most of the members participate also into other development activities of the same system, it guarantees that quality modeling will be integral and dominant part of the whole software development process. The actor-informant group is properly formed if all essential actor-stakeholder categories are represented. Case studies showed, however, that it is often difficult to engage all relevant stakeholders. Therefore, it is practical to start the work with most immediate system users and expand the informant group later according to needs and possibilities. If relevant, cultural aspects have to be taken into account when actor group is formed. Some studies in developing countries (e.g. Thanasankit and Corbitt, 2000, using a Thai case) show that social structures and hierarchies can be tall. All kinds of decisions must be approved by managers or committees. In these kinds of environments actor group must cover not only people with knowledge but also people with power. Before starting the actual quality modeling the informant group must be aware of the perspectives (step 1a) its members possess and the views (step 1b) they can have of the target system and its context. After that comes the task of gaining initial understanding of system and context.

The information system can first be described (step 2a) to actors more or less as a "black-box" or by simple structural diagrams. The purpose of this view is to turn attention from the very beginning to the desired relationships between system and its context, i.e. quality requirements. It resembles the Taylorian view where, for example, messages stored in information system have no inherent value, and the value of entire system emerges only within a context (see Scholl et al. 2011, 790). The description of information system will gradually become more detailed and transparent during "attribution" and "contributors" steps. After initial system description follows the initial description of the system context (step 2b). It is a more important task in the initial phase than description of the system itself. Entities in the environment, including different human actors, determine what is required of the relationships between system and context. *A* suitable "meta-model" or theory of context

would guide in focusing on the most relevant features of environment with respect to quality modeling.

After inception phase the overall quality design can start. It consists of creating a prioritized system specific set of domains (areas of concern) and quality attributes out of known alternatives. The results of inception phase together with the initial overall quality design form a kind of sketch of the target system in relation to context. In one of the case studies actors were given a large combined selection of domains and attributes and asked to prioritize them. This was found to be difficult for some informants because of too many items to deal with. Therefore, in the following case studies domain collection was separated from the "palette" and actors were asked to prioritize the domains (step 3) and attributes separately. During overall quality design conflicts may arise and a method for solving them must be found.

Case study observations suggest that after letting actors prioritize domains the most useful and productive step is eliciting positive and negative facts (step 4) indicating existence or the lack of quality inside each prioritized domain. This starts the detailed quality design. In the meta-model these facts are called "indicators" and they constitute an important part of individual quality attribute models. The identification of use cases, scenarios (step 5) and individual quality attributes (step 6) related to the facts can follow thereafter, as well as prioritizing the attributes. Next comes looking for contributors (step 7) and possibly attributing (step 8) the qualities more precisely to certain system constituents. In these steps actors need a lot of support from someone experienced in quality modeling. The assumption that stakeholders are able to understand and communicate present and future needs in a clear way has been recently criticized for example by Holmström and Sawyer (2011, 35). Defining internal contributors is part of system model and consequently requires that developers take part in the process. The steps from 5 to 8 are in practice carried out rather simultaneously than one after another. Unless system component or sub-system specific attribute collections and sets are used, quality characteristics are usually at the beginning of quality modeling attributed to the information system as a whole. More specific attributions grow up during the modeling process, especially in connection with contributor element. All the steps of detailed quality design can potentially affect and cause changes in the overall quality design, i.e. the prioritized domain-attribute set.

After listing indicators and identifying use cases, scenarios and contributors there exists enough material for creating a formal representation each desired relationship between system and context called "model" (step 9). At this stage indicators, model and its verbal description can be used to refine the general definition of the attribute in question in domain-attribute set. The remaining steps are identification of attribute relationships (step 10), especially conflicts, and designing a procedure for measuring (step 11) actual attribute values in connection with testing and operating the target system.

The principles of flexibility and freedom are important in the selection of system specific domain-attribute sets and in quality-driven development in general. There are, however, some core requirements that are commonly acknowledged to be important per se and should always be included in attribute sets. First of all any information system must be feasible (before even trying to create or acquire it), available, accessible and sustainable. In addition, it must be useful and therefore frequently used or, in some cases (e.g. computer games), have an ability to entertain. All the other characteristics, usability in the front line,

follow from or affect the before mentioned. They hamper or make it easier to use the information system, make it less accessible etc. Figure 12 depicts this view.

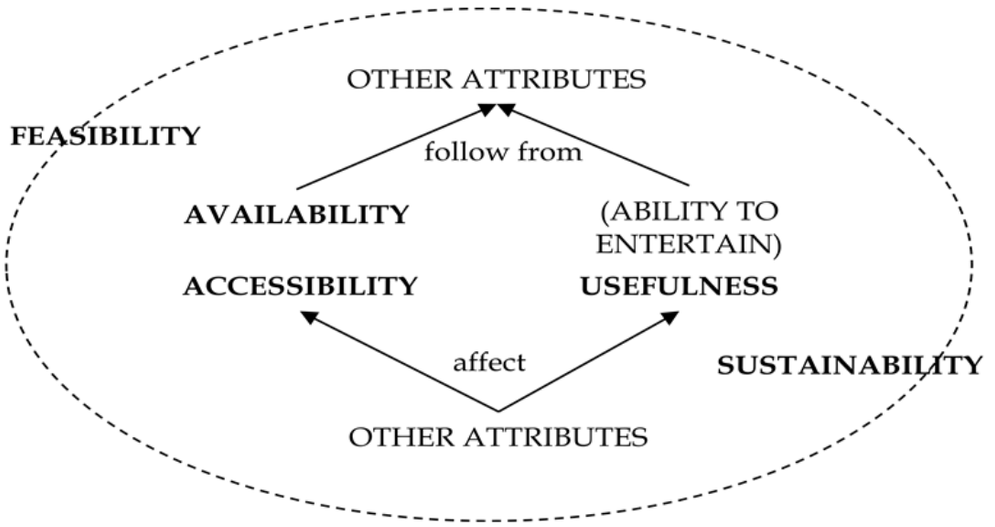


Fig. 12. Core quality requirements

The division into core qualities and other qualities resembles the division into key quality factors and locally defined factors by Khaddaj and Horgan (2005) in their Adaptable Quality Model (AQM). The key factors are required of all products. Locally defined factors, in turn, apply only to the current product being developed. AQM defines in total seven key qualities: maintainability, usability, cost/benefit, security, reliability, timeliness and correctness. Compared to AQM Figure 12 represents even more fundamental requirements. In a recent article Buschmann (2011), in turn, underlines just usefulness (in his terminology "business suitability") and usability as the key requirements for software.

5. Conclusion

The above sections have given a definition for quality with respect to information systems as technical artifacts. In addition they have presented a holistic conceptual framework for quality modeling and a way to apply it in the course of system development. The last section finally promoted quality model to arch-model of an information system. As a theory, the quality meta-model does not assert anything testable about the relationships between its elements. The most relevant method of evaluating the framework is assessing the actual quality models created by applying it. These system specific models must be useful, contain elements that represent the real world and be comprehensible. In addition the meta-model itself must be comprehensive, flexible, general enough and applicable to a variety of contexts. This entails repeated case studies. While the framework does not offer testable propositions, it opens a number of questions for future research. How to arrange quality attributes into categories? Can a widely acknowledged division into domains be achieved? Are some actor perspectives and views more informative and productive than others? What are the most relevant axes in the frame of reference that determine the appearance of system to an actor? How is the physical, infrastructural and cultural context reflected in quality mnodels?

6. References

Allen D., Karanasios S., Slavova M. (2011). Working With Activity Theory: Context, Technology and Information Behavior. *Journal of the American Society for Information Science and Technology. 62(4),* 776-788.

Alter S. (1999). A General, Yet Useful Theory of Information Systems. *Communications of the Association for Information Systems. 1(3),* Art. 13.

Alter S. (2008). Defining information systems as work systems: implications for the IS field. *European Journal of Information Systems, 17(5),* 448-469.

Buschmann F. (2011). Unusable Software Is Useless, Part1. *IEEE Software, 28 (1)*, 92-94.

Finne A. (2006). Applying a Twofold Quality Model: Producing Groundwork for System Specific Attribute Models. In Hautop H., SutinenE., Duveskog M., Kinshuk, Mkocha A. (eds.) *Proceedings of the Fourth IEEE International Workshop on Technology for Education in Developing Countries, Iringa, Tanzania, July 10-12, 2006,* 3-4.

Finne A. (2011). Towards a Quality Meta-Model for Information Systems. *Software Quality Journal, 19(4),* 663-688.

Homström J., Sawyer S. (2011). Requirements engineering blinders: exploring information systems developers' black-boxing of the emergent character of requirements. *European Journal of Information Systems, 20(1)*, 34-47.

IEEE (Institute of Electrical and Electronics Engineers) Standard Glossary of Software Engineering Terminology (1990). New York: IEEE.

ISO (International Organization for Standardization) 8402 (1994). *Quality management and quality assurance – Vocabulary*. Geneva: ISO/IEC.

ISO (International Organization for Standardization) 9000 (2000). *Quality management systems – Fundamentals and vocabulary*. Geneva: ISO/IEC.

ISO (International Organization for Standardization) 9126 (2001). *Software Engineering – Product Quality – Part 1: Quality Model*. Geneva: ISO/IEC.

ISO (International Organization for Standardization) 12207 (2008). *Systems and software engineering – Software life cycle processes*. Geneva: ISO/IEC.

Jacobson I., Booch G., Rumbaugh J. (1999). *The Unified Software Development Process.* Boston: Addison-Wesley.

Khaddaj S., Horgan G. (2005). A Proposed Adaptable Quality Model for Software Quality Assurance. *Journal of Computer Sciences, 1(4)*, 482-487.

Kotonya G., Sommerville I. (1996). Requirement engineering with viewpoints. *IEEE Software Engineering Journal, 11(1),* 5-18.

Macome E. (2008). On Implementation of an Information System in the Mozambican Context: The EDM Case Viewed Through ANT Lenses. *Information Technology for Development, 14(2),* 154-170.

Muhren W., van den Eede G., van de Walle B. (2008). Sensemaking and Implications for Information Systems Design: Findings From the Democratic Republic of Congo's Ongoing Crisis. *Information Technology for Development, 14(3),* 197-212.

Narasimhaiah G., Lin S-L. (2010). Determinants of software quality: A survey of information systems project managers. *Information and Software Technology 52 (6)*, 602-610.

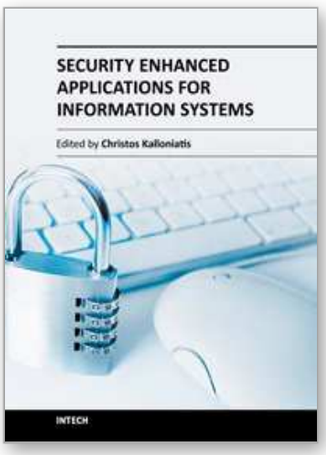
Reeves C., Bednar D. (1994). Defining quality: Alternatives and implications. *The Academy of Management Review, 19(3),* 419-445.

Savolainen R. (2006). Information use as gap-bridging: The viewpoint of sense-making methodology. *Journal of the American Society for Information Science and Technology, 57(8),* 1116-1125.

Scholl H., Eisenberg M., Dirks L., Carlson T. (2011). The TEDS Framework for Assessing Information Systems From a Human Actor's Perspective: Extending and Repurposing Taylor's Value-Added Model. *Journal of the American Society for Information Science and Technology, 64(4),* 789-804.

Silva L. (2007). Institutionalization Does Not Occur By Degree: Institutional Obstacles in Implementing a Land Administration System in a Developing Country. *Information Technology for Development, 13 (1),* 27-48.

Thanasankit T., Corbitt B. (2000). Cultural Context and its Impact on Requirements Elicitation in Thailand. *The Electronic Journal on Information Systems in Developing Countries, 1(2),* 1-19.



Security Enhanced Applications for Information Systems

Edited by Dr. Christos Kalloniatis

ISBN 978-953-51-0643-2
Hard cover, 224 pages
Publisher InTech
Published online 30, May, 2012
Published in print edition May, 2012

Every day, more users access services and electronically transmit information which is usually disseminated over insecure networks and processed by websites and databases, which lack proper security protection mechanisms and tools. This may have an impact on both the users' trust as well as the reputation of the system's stakeholders. Designing and implementing security enhanced systems is of vital importance. Therefore, this book aims to present a number of innovative security enhanced applications. It is titled "Security Enhanced Applications for Information Systems" and includes 11 chapters. This book is a quality guide for teaching purposes as well as for young researchers since it presents leading innovative contributions on security enhanced applications on various Information Systems. It involves cases based on the standalone, network and Cloud environments.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen