Chapter 8
Relational Service Quality Modeling

Vladimir A. Shekhovtsov
National Technical University "Kharkiv Polytechnic Institute", Ukraine

Roland Kaschek
Information Science Research Center, New Zealand

Christian Kop
Alpen-Adria-Universität Klagenfurt, Austria

Heinrich C. Mayr
Alpen-Adria-Universität Klagenfurt, Austria

ABSTRACT

The paper argues that using non-first-normal-form (NF^2) tables for requirements modeling is a suitable approach for communicating application system issues to stakeholders who have a business background. In particular, such tables are used in an intermediate predesign step residing between requirements elicitation and conceptual design for modeling functional and quality requirements of services and business processes. This approach extends to quality requirements of business processes used for service orchestration.

CONTEXT AND MOTIVATION

Our work focuses on the elicitation and modeling of service and business process quality requirements within the context of Service-Oriented Architecture and Process-aware Information Systems (PAIS) (Dumas, Van der Aalst, & ter Hofstede, 2005) that are facilitated by computer applications such as BPM engine (workflow management system).

Requirements concerning system quality are often referred to as *non-functional requirements* in order to distinguish them from what traditionally is just called *requirements* in the realm of information systems and only covers functional system aspects. We consider quality requirements

DOI: 10.4018/978-1-60566-794-2.ch008

regarding a system under development (SUD) as a specification of the quality that the SUD is expected to have in order to be fit for use. Thus, featuring the right functionality, i.e. implementing the functional requirements, might be one quality aspect among others. Within the context of SOA, quality requirements are related to both individual services and orchestrations thereof, i.e. business processes.

Elicitation and modeling of quality requirements is important and challenging because:

1. System development is often driven by stated requirements rather than stakeholder needs; inappropriate quality requirements likely are going to waste resources spent for development.
2. Requirements engineering is about system branding, i.e., conceptually shaping or working out the related SUD. Using abstract design-time notions (classes, attributes, activities etc.) at that stage of the development lifecycle for describing system qualities might compromise that branding, as the chosen concepts might be inappropriate for stakeholder understanding and validation.
3. In future, automated service selection, based on automated negotiations, is likely to take place. These negotiations involve service quality characteristics such as performance, security, cost, risk, and similar. That approach obviously has the potential to fundamentally change the way web applications are created.

Our approach is built on the predesign technique described in (Kop & Mayr, 2002; Mayr & Kop, 1998; Shekhovtsov, Kop, & Mayr, 2008) which is about modeling functional requirements for information systems. Our approach aims at quality requirements as opposed to functional ones, and focuses on services and business processes. For empowering stakeholders to cope with service and business process quality issues we propose using a simple but sufficiently expressive semantic model for addressing the relevant concepts. That model is represented as a *non-first-normal-form* (NF^2) relational model (Schek & Pistor, 1982) permitting repetition groups and nested relations. In the present context we call a NF^2 table *glossary*. Its advantage is that it allows for the specification of any relationship between any number of given concepts. In mathematics such a definition of concepts in terms of each other and relationships among them is called an *implicit definition* or *axiomatization*.

Experience from practice suggests that stakeholders having a business background with relative ease understand predesign models because they come along in the well-known shape of tables (Galle, Kop, & Mayr, 2008; Janicki, Parnas, & Zucker, 1997). The conceptual predesign model also is suitable for model mapping. For static and dynamic aspects it was shown that predesign modeling notions can be mapped into class diagrams, activity diagrams and state charts (Kop & Mayr, 2002; Mayr & Kop, 1998) or modeling notions of business process models can be derived from them (Mayr, Kop, & Esberger, 2007). The aim is to guarantee such a mapping flexibility also for quality requirements. Our approach, as suggested by (Mayr, 2006; Pastor, 2006), thus enables both: focusing on requirements and stakeholders' needs, and model mapping onto design notation such as UML and further onto executable code.

In addition, using a unified tabular (relational) representation for requirements allows users to perform relational queries over requirements repositories using a domain-specific SQL-like language. This will be illustrated by examples later in the chapter.

The chapter is organized as follows: in the next section we discuss the foundations of our work and rationalize in favor of our approach. Then we apply our approach to capture service and process quality requirements. The paper is completed with a discussion of the related work and a conclusion.

MODELING SERVICE AND BUSINESS PROCESS QUALITY

We consider quality of an artifact as that artifact's fitness for a specified kind of use. Thus, when a particular kind of use is considered we aim at defining quality characteristics which aid in discussing that particular fitness for use. Our quality requirements model therefore builds on the concept of service or business process quality.

For working out a conceptual framework of requirements models we briefly look into quality requirements metamodels. The paper (Jureta, Herssens, & Faulkner, 2008) has suggested a related metamodel. We reuse that paper's quality requirements metamodel, add a number of metamodel characteristics that have been missed out by that paper, and adapt the terminology to our purposes. We call these quality requirement metamodel characteristics *quality model parameters* because they affect the quality model without being an explicit part of it.

We identify the following quality model parameters:

1. **Context:** the considered circumstances of a service or process execution request.
2. **Dependency:** the permitted relationships between different quality characteristics.
3. **Priority:** the importance of quality characteristics.
4. **Schema:** the kind of organization of the quality characteristics (taxonomy, ontology etc.).
5. **Scoring:** the concepts and rules for scoring requirements.
6. **Stakeholder:** the stakeholders whose requirements are to be considered.
7. **Status:** a given requirement model's life cycle phase.
8. **Polymorphy:** whether or not a quality requirement's metamodel is fixed.
9. **Version:** a given requirement model's version number.

Service and Business Process Quality Models

To describe service quality issues, the concept of Quality of Service (QoS) is widely used. It roots in the research on network-related service delivery performance and reliability. Currently it is usually understood in broader sense – as a synonym for service quality in general.

Service and Business Process Quality Taxonomies

The number of service quality models that employ a taxonomy of quality characteristics (possessing the *schema* parameter value of taxonomy) is quite large and we discuss only a small subset of them. Most taxonomy-based models addresses QoS (Dobson, 2004; Evans & Filsfils, 2007). The models in (Chaari, Badr, Biennier, BenAmar, & Favrel, 2008; Galster & Bucherer, 2008; Ran, 2003) utilize QoS. (Al-Masri & Mahmoud, 2008; Andreozzi, Montesi, & Moretti, 2003; O'Sullivan, Edmond, & ter Hofstede, 2002; Zeng et al., 2004) refer to a general notion of quality with application to services. We will not make further distinctions between these two categories of models.

A number of efforts address the standardization of service quality models. For example, OASIS published the Web Services Quality Model (WSQM) (E. Kim & Lee, 2005; Lee & Yeom, 2007). It is intended to serve all the associated bodies. There are models utilizing existing quality model standards, such as the SQuaRE series of quality standards (ISO/IEC 250xx) (Abramowicz, Zyskowski, Suryn, & Hofman, 2007). In our further examples, we will use WSQM as an example of comprehensive service quality model. The approach of (Lee & Yeom, 2007) adds the concept of quality chain to WSQM for addressing *dependency* quality model parameter. The *stakeholder* parameter is addressed by WSQM as it allows for detailed treatment of the abilities of stakeholders to express different quality requirements.

Most of the quality models for business processes are taxonomies of quality characteristics (Aburub, Odeh, & Beeson, 2007; Guceglioglu & Demirors, 2005). Low-level quality characteristics are accompanied with metrics (Lee, Bae, & Shin, 2005; Vanderfeesten, Cardoso, Mendling, Reijers, & van der Aalst, 2007).

Mostly, regarding *polymorphy,* the value *fixed* is employed, i.e. the resp. models do not allow to define additional characteristics or metrics. (Liu, Ngu, & Zeng, 2004) propose a model which employs the parameter value *float*. It permits to obtain numerical scores for the requirements model quality. Changing the requirements model metamodel is guaranteed by uniformity of QoS calculation which does not depend on particular quality characteristic.

Service Quality Ontologies

Quality can also be conceptualized as an ontology. Most related approaches draw from semantic web services concepts; we will describe them together with a discussion of the possible mapping of our predesign model into such conceptualization in the "related work" section. In this section, we briefly mention several ontologies not directly connected to the Semantic Web.

A set of requirements for two unified ontologies: specific QoS ontology and Service-Level Agreement ontology (including, among others, notions for QoS requirements) is presented in (Dobson & Sanchez-Macian, 2006). An ontological approach for QoS representation introduced in (Zhou, Niemelä, & Savolainen, 2007) is based on two levels of ontologies: an upper-level QoS ontology introducing such domains as entity, people, process, property and means and lower-level ontologies for these domains (with an example of QoS property lower-level ontology actually conceptualizing the quality model). Another ontology representing QoS requirements (subdivided into requirement, measurement, traceability, and quality management system ontologies) is presented in H. M. Kim, Sengupta, and Evermann (2007).

PREDESIGN CONCEPTS

The proposed technique merges two mutually supplementing approaches: an approach to model functional requirements for services and business processes and an approach to model quality requirements: In this section, we outline the common properties of these approaches; approach-specific concepts will be covered in the subsequent sections.

Predesign Process

We propose a five step *predesign process* for requirements elicitation and modeling. Note that the first two steps are drawn from requirements engineering and only the remaining steps are predesign specific. Therefore we will not step into much detail of the first two steps but concentrate on the remaining ones (3 to 5) instead. Note that not all approaches to perform Step 3 depend on performing the previous steps. The predesign process steps are supposed to be iterated, as long as no more need for change is felt:

Step 1: Identify the relevant actors, tasks and quality concerns. Before any elicitation of requirements can start the right *tasks* and *actors* have to be decided upon. The identification of actors is an iterative task itself since the requirements engineer typically has to start with persons nominated by management and then to find out other relevant people, i.e., stakeholders who might provide relevant information about important tasks or for a task at hand: e.g., persons who are interested that the project will become a success, opinion leaders for the project, or persons affected by the execution of the task. During each interview, the meta question should be asked: "whom else can I interview to certain aspects of what you have told me?" thus establishing iteration. Among others this is a simple rule of thumb to get in contact with candidates for actors.

Examples of actors are humans and organizational units. Actors and tasks will later be related to each other by the association "is responsible for". We provide special tables, in particular "organizational unit/task" and "task/information provider" to help in organizing this step.

Step 2: Establish a requirements elicitation plan. For questioning actors, a kind of schedule and guideline should be worked out. This, in our opinion, will be quite project specific, nevertheless, it is good practice to start at the management level going to the operative level next. Managers can draw an overview of the planned system's aims and concerned people although will not utter many operative details (functionality of the system). Besides, early decision maker participation makes them feeling responsible which may substantially strengthen the requirements engineer's position.

Furthermore it has to be decided on requirements priorities from indispensable to "nice to have". This will help to avoid getting lost in detail discussions about unnecessary features.

Step 3: Collect detailed requirements according to the elicitation plan. The conventional technique is to elicit requirements via interviews or brainstorm sessions. This way, free-form or structured answers to interview questions will be obtained as raw input data for predesign. In addition to that even entire free-text specifications may be used and transformed (semi)- automatically into glossary entries using natural language processing techniques (see, e.g., (Fliedl, Kop, Mayerthaler, Mayr, & Winkler, 2002)). All entries will be indexed by task, stakeholder, organizational unit, date, etc., and, if necessary, manually revised (finding implicit information, eliminating redundancies, etc.).

Collecting requirements based on specification texts is convenient only if these texts can be obtained easily. Concerning quality requirements we have to encounter that the known problem stakeholders usually have with uttering requirements without experiencing the system even grows. Without such experience they are forced to be speculative and, as a result, formulate requirements that need to be further refined and corrected. To solve this problem (Kaschek, Kop, Shekhovtsov & Mayr, 2008) proposed a tool-supported approach providing stakeholders with a means for experiencing software qualities in a simulated working environment as early as possible, and assessing these simulated qualities using some defined scale. This approach will also provide requirements engineers with a means for eliciting quality requirements out of such assessments.

Step 4: Organize the elicited requirements. We propose a specific **predesign model** to organize the elicited requirements. It consists of a set of **predesign glossaries** representing a small set of modeling concepts that are intuitively understandable and verifiable by stakeholders; this is enhanced by the tabular glossary representation which is well-known to our intended audience, i.e., business people. After verification, the requirements glossaries have to be mapped onto a design-time representation (predesign mapping step) which will be discussed in detail in this chapter.

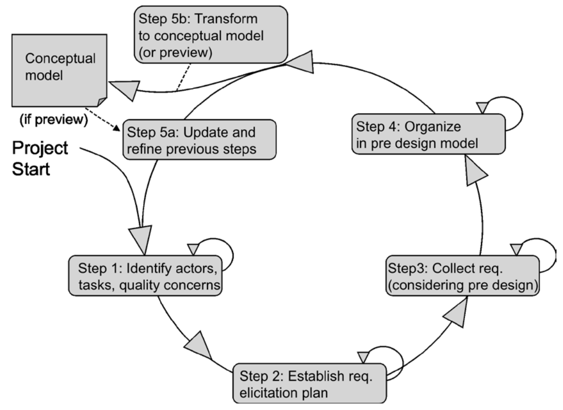
Step 5: Transformation to conceptual schema or update of actors, tasks, requirements, and requirements elicitation plan. According to whatever respective need is found during steps 1 to 4 the requirements collected in the predesign model are, if necessary, validated, updated and refined, and finally transformed to a conceptual schema. Note, that this process might be iterative in case of more sophisticated information systems.

A graphical summary of the process can be seen in Figure 1.

Predesign Metamodeling Hierarchy

The predesign process runs along a model hierarchy consisting of three layers: object-model (M1),

Figure 1. The predesign process


Conceptual model
Step 5b: Transform to conceptual model (or preview)
(if preview)
Step 5a: Update and refine previous steps
Project Start
Step 4: Organize in pre design model
Step 1: Identify actors, tasks, quality concerns
Step3: Collect req. (considering pre design)
Step 2: Establish req. elicitation plan

meta-model (M2) and NF^2 or meta-metamodel (M3).

The object model layer allows defining particular models that describe the requirements for the case at hand. These predesign models consist of a small set of concepts, can be verified by stakeholders and later mapped into design. The predesign concepts represent things in a real world or their attributes or operations etc. Higher in the metamodel hierarchy, the predesign model is defined via a predesign metamodel which is an object-oriented depiction of predesign concepts (a concept corresponds to a metaclass). On top, the predesign meta-metamodel layer describes the way of representing the metamodeling structures. We propose to use the Non-First-Normal Form (NF^2) relational notion for this layer.

We define a predesign glossary as a NF^2-table view over our metamodel populated with data. This approach is valid due to the following observations (along the lines of Codd, 1979):

1. Within the relational context, any object identity can be modeled by a tuple identifier.
2. Object methods can be modeled by tables that on request will be joined with the table holding the object's basic data.
3. Each of the metamodel diagrams can have a relational representation.

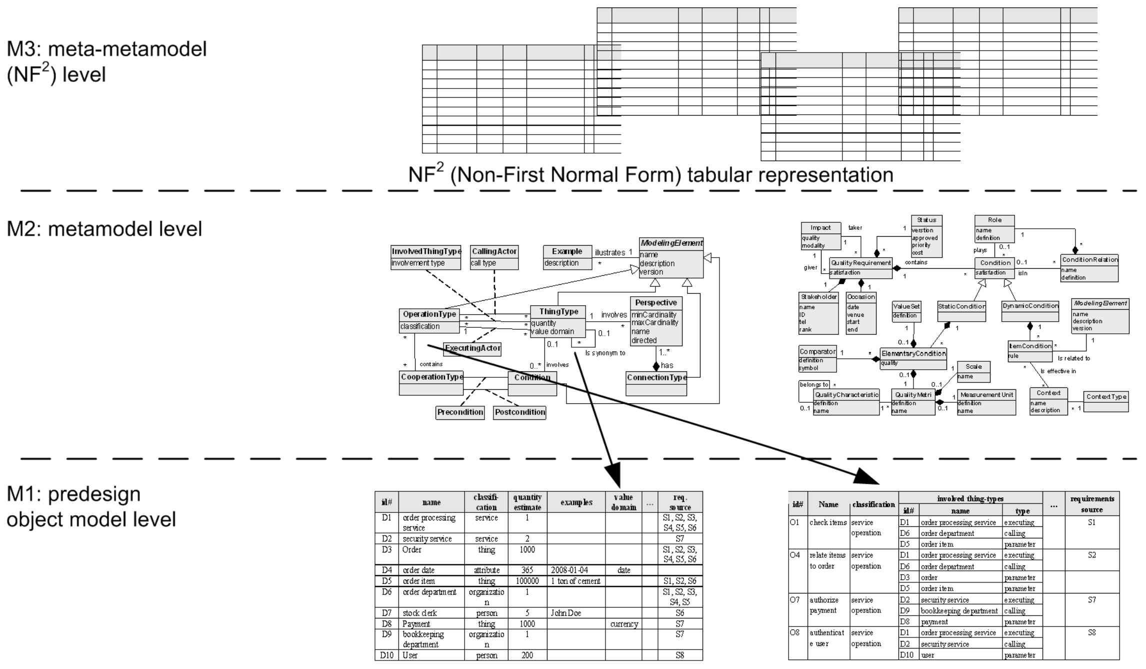
Different predesign glossaries will be defined in detail subsequently.

Example Requirements Specification

Our further description of predesign techniques will be based on a simple example of both service and business process requirements specifications which include functional and quality requirements. For brevity, implementation constraints are not included, and the number of requirements is kept rather small. We list the requirements for a set of order processing services: (Figure 2).

1) Order processing services should allow the order department to perform:
   (S1) order acceptability check;
   (S2) relating the items to the order;
   (S3) order confirmation;
   (S4) order rejection; and
   (S5) order postponing.
2) Order processing services should allow the stock clerk to perform:
   (S6) stock item top-up;
3) Authorization services should allow the bookkeeping department to perform:
   (S7) payment authorization;
4) Authorization services should allow the order processing services to perform:
   (S8) user authorization check.
5) The order processing services must have the following qualities:
   (S9) The response time to user actions related to orders and order items is below 0.5 sec.
   (S10) The response time to any payment authorization is below 0.3 sec.
   (S11) The response time to any stock items top-up for dialup connection requests is below 2 sec for a load up to 500 users and below 5 sec for a higher load; it is below 0.5 sec for all other requests.

Figure 2. Requirements displayed on predesign object model, metamodel and NF² meta-metamodel level



id#	name	classifi-cation	quantity estimate	examples	value domain	...	req. source
D1	order processing service	service	1				S1, S2, S3, S4, S5, S6
D2	security service	service	2				S7
D3	Order	thing	1000				S1, S2, S3, S4, S5, S6
D4	order date	attribute	365	2008-01-04	date		
D5	order item	thing	100000	1 ton of cement			S1, S2, S6
D6	order department	organizatio n	1				S1, S2, S3, S4, S5
D7	stock clerk	person	5	John Doe			S6
D8	Payment	thing	1000		currency		S7
D9	bookkeeping department	organizatio n	1				S7
D10	User	person	200				S8

id#	Name	classification	involved thing-types			...	requirements source
			id#	name	type		
O1	check items	service operation	D1	order processing service	executing		S1
			D6	order department	calling		
			D5	order item	parameter		
O4	relate items to order	service operation	D1	order processing service	executing		S2
			D6	order department	calling		
			D3	order	parameter		
			D5	order item	parameter		
O7	authorize payment	service operation	D2	security service	executing		S7
			D9	bookkeeping department	calling		
			D8	payment	parameter		
O8	authenticat e user	service operation	D1	order processing service	executing		S8
			D2	security service	calling		
			D10	user	parameter		

(S12) Any operation performed by the order department has highest possible availability.

(S13) Stock item top-up has second highest availability.

(S14) Only authorized users can use the order processing service.

6) The business process for order processing is defined as follows (Fowler, 2005):

(S15) The order comes in.

(S16) The order department for each ordered item checks its availability on stock.

(S17) If each ordered item is on stock, then the order department relates that item to the order.

(S18) If the item quantity on stock is below the threshold, then the stock clerk orders this item for the stock.

(S19) If the order comes in, the bookkeeping department checks the payment.

(S20) If the payment is authorized and all ordered items are on stock, then the order department confirms the order.

(S21) If payment is authorized and some items are not on stock, then the order department marks the order as pending.

(S22) If the payment is not authorized, then the order department must reject the order.

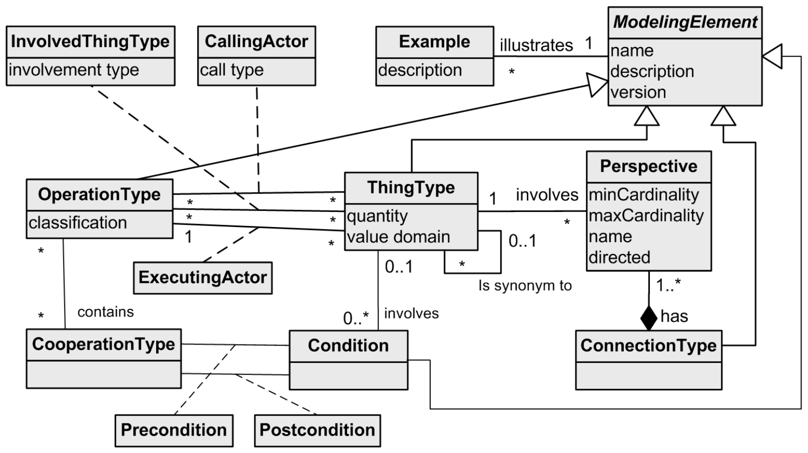
7) The business process for order processing must have the following qualities:

(S23) The availability of all the operations performed if order comes in must exceed 99%.

(S24) The activities performed if the order is rejected must have the response time below 0.3 sec.

In the next two sections, we will describe how the proposed predesign technique allows for modeling functional and quality requirements. These two parts of the predesign process must be performed together to form the common predesign model using the appropriate metamodel. We will describe the metamodels for these two kinds of

Figure 3. Part of the predesign metamodel describing the model for functional requirements



requirements separately but actually they are both parts of the same predesign metamodel.

RELATIONAL PREDESIGN OF FUNCTIONALITY

The predesign technique for functionality presented in this paper is based on a technique described in (Kop & Mayr, 2002; Mayr & Kop, 1998). The corresponding part of the predesign metamodel is shown on Figure 3.

The most general concept of functionality predesign is *ModelingElement*. It permits referring to any phenomena and, among others, is specialized into thing-type and connection-type. The concept *ThingType* generalizes conceptual notions such as class, entity, attribute, or value. The instances of the concept *ConnectionType* are relationships between instances of thing-types. Any such relationship is considered as a set of *Perspectives*. Each perspective of a relationship models the contribution instances of the resp. involved thing-type have in that relationship. Typical instances of thing-types are natural or juristic persons, material or immaterial objects, abstract notions. In textual requirements specifications they are usually referred to by noun phrases.

Predesign models include a *statics* and a *dynamics* submodel, having operation invariant and dependent information, respectively. Thing-types and connection-types constitute the statics predesign submodel. The dynamics submodel consists of *OperationTypes* (defining the permitted operations) and *CooperationTypes* including pre and post-conditions for operation execution (Kop & Mayr, 2002; Mayr et al., 2007).

The metamodel includes (among others) the “is synonym to” link permitting the use of synonyms. Examples can be specified for all predesign concepts; this is represented via “illustrate” link. Concept versioning is supported with a “version” attribute of ModelingElement.

Glossary-Based Modeling Using Thing-Types and Connection-Types

First two kinds of predesign glossaries are thing-type and connection-type glossaries. They are NF^2 representations of the metamodels for the corresponding predesign notions.

We need to introduce thing-types and connection-types present in our domain, as we will use these types throughout the example model. The corresponding glossaries are shown on Table 1.

Table 1a. Thing-type and connection-type

id#	name	classification	quantity estimate	examples	value domain	req. source
D1	order pocessing service	service	1			S1, S,S3, S4, S5, S6
D2	security service	service	2			S7
D3	Ordr	thing	1000			S1, S, S3, S4, S5, S6
D4	order date	attribute	365	2008-01-04	date	
D5	order iem	thing	100000	1 ton o cement		S1, S2, S6
D6	order department	organization	1			S1, S2, S3, S4, S5
D7	stock clerk	person	5	John Do		S6
D8	Payment	thing	1000		currec	S7
D9	bookkeeping department	organization	1			S7
D10	User	person	200			S8

Using Operation-Types to Model Service Functional Requirements

Following (Mayr et al., 2007) we model software services using the concept of operation-type. The corresponding metamodel is shown in Table 2. Every operation-type is related to its executing actor, originating actor(s), and resources such as operation subject and auxiliary or associated materials (operation parameters). These are referred to as involved thing-types.

The requested operation-types are to be collected into an *operation-type glossary* (see Table 2).

Using Cooperation-Types to Model Business Process Functional Requirements

The business process functionality is the set of permitted service flows as specified via the business process definition.

Following (Mayr et al., 2007) our organization modeling concepts are: *Organization*, the whole entity such as a tax office or municipal administration; *Department* and/or *Project Group*, i.e., functioning subunits of an organization; *Position*, atomic organization subunits which are occupied by a set of actors; *Citizen*, *Customer*, *Supplier* – organization external actors interacting with

Table 1b. Thing-type glossary examples

c-id#	name	…	Perspective					requirements source
			p-id#	involved thing-type	name			
C001	containent		p001a	D3, order	contains			S1, S2, S6
			p001b	D5, order item	is contained in			
C002	attribute possessing		p002a	D3, Order	has an attribute			
			p002b	D4, order date	is an attribute of			

Table 2. Example operation-type glossary

id#	Name	classification	involved thing-types			requirements source
			id#	name	type	
O1	check iems	service operation	D1	order processing service	executing	S1
			D6	order department	calling	
			D5	order item	parameter	
O4	relate items to order	service operation	D1	order processing service	executing	S2
			D6	order department	calling	
			D3	order	parametr	
			D5	order item	parameter	
O7	authorize payment	service operation	D2	security service	executing	S7
			D9	bookkeeping department	calling	
			D8	payment	parameter	
O8	authenticate user	service operation	D1	order processing service	executing	S8
			D2	security service	calling	
			D10	user	parametr	

it; *Resource* and *Document*, used during task execution. They are materialized as thing-type instances. Any such instance providing or using a service occupies the role of actor executing or initiating that service, respectively. In addition, we use the concepts of *Task* and *Service Flow*, i.e., elementary and composite activity in favor of the organization's goals. Any service flow of an organization is described as network of tasks each of which is executed by a position. Any task may have inputs and outputs.

To generalize service flow modeling notions, (Kop & Mayr, 2002; Mayr et al., 2007) introduce the concept cooperation-type. A *cooperation-type* C is a triple (–C, C(O), C+) of pre- and post-condition sets –C and C+, respectively, and the set C(O) of operation-types contained in C. For two cooperation-types C and D the *concurrent composition* of operation-types C(O) and D(O) is defined for –C = –D and C+ = D+. The *sequential composition* C(O) before D(O) is defined for C+ = –D.

Tasks and positions are modeled as operation-type and task executing actor, respectively. To any task set there is thus associated a set of pre- and post conditions, respectively. Task inputs and outputs are modeled as thing-types related to operation-types as parameters. Any business process phases are modeled as cooperation-types. An excerpt from a cooperation-type glossary is shown on Table 3. Among other tabular requirements modeling approaches, the closest counterpart of this glossary is a mode transition table used in SCR method (Heitmeyer, 2007).

RELATIONAL PREDESIGN OF QUALITY

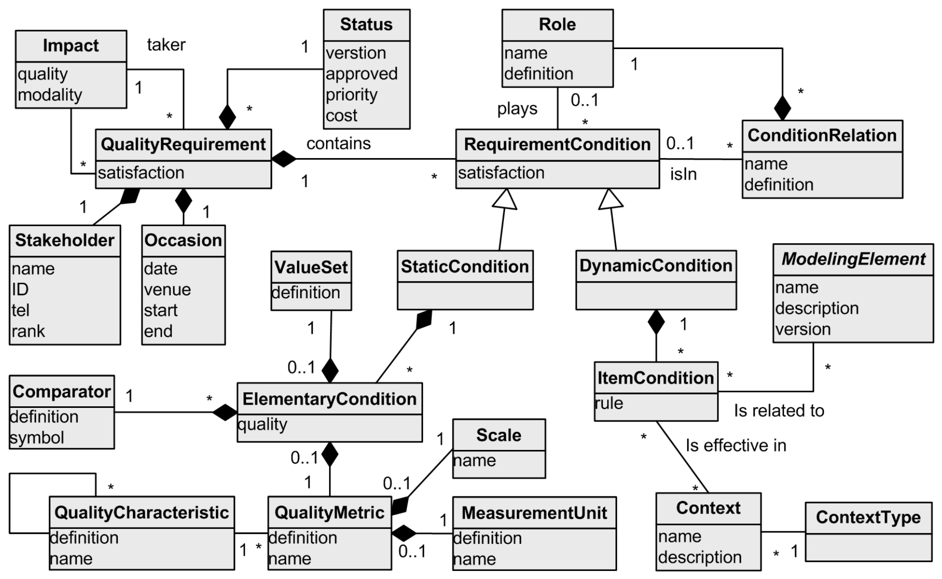
A part of the common meta-model describing the predesign model for service and business processes quality requirements is shown on Figure 4. For obtaining it we have used the papers (Aburub et al., 2007), (Kassab, Ormandijeva, & Daneva, 2008), and (Jingbai, Keqing, Chong, & Wei, 2008).

A quality requirement is essentially an utterance declaring that something should be in a particular and specified way. The metamodel depicts this via the metaclass *QualityRequirement*. Quality requirements may impact each other. Each

Table 3. Example cooperation-type glossary

id#	Pre-condition			Operation			Post-condition			req. source
	id#	name	involved types	id#	name	involved types	id#	name	involved types	
E1	C1	order comes in	D3, Order	O7	check payment	D2, security service D9, bookkeeping department D8, Payment	C2	payment is authorized	D8, Payment	S15, S19
							alternate			
							C3	payment is not authorized	D8, Payment	
				in parallel						
				O1	check all items	D1, order processing service D6, order department D5, Order	C4	all articles are in stock	D3, Order	S15, S16
							alternate			
							C5	not all articles are in stock	D3, Order	
E2	C3	payment is not authorized	D8, Payment	O5	reject order	D1, order processing service D9, bookkeeping department D3, Order	C6	order is rejected	D3, Order	S22

Figure 4. Part of the predesign metamodel describing the model for quality requirements



such impact has a number of givers and takers. The attributes of an impact are quality and modality. Quality means supporting or hindering to achieve the satisfaction of the quality requirement. Modality means the extent to which achieving of that requirement's satisfaction can be supported or hindered. A quality requirement's satisfaction can be scored as a value in {0,1}, [0,1], or different. For simplicity we avoid including into that metamodel a facility for specifying the escalation of requirements satisfaction from sub-requirements to super requirement and vice versa. Quality requirement is a composite of *Stakeholder* (representing the person responsible for verifying the requirement), *Occasion* (representing the information related to the specific case of eliciting the requirement), the requirement *Status* (its meta-attributes include version, priority reflecting its importance, cost of non-meeting the requirement reflecting its urgency etc), and *Requirement-Condition*, i.e., the one which is actually requested by the requirement.

We propose to employ two kinds of condition: static and dynamic ones. Furthermore, conditions can be in *Relations* to other conditions. A condition relation is defined as the composite of a number of *Roles*. If a condition is in a relation to another condition that it can play (in that relation) only play roles that are defined for that relation.

Dynamic Conditions: Connecting Quality and Functionality

At the instance level, dynamic conditions include a number of *functional item* conditions connecting this requirements with artifacts defined in the functionality-related part of the predesign model. We consider three kinds of functional items: *activity* (represented via operation-type), *event* (represented via cooperation-type), and *actor* (represented via thing-type). These three kinds of conditions allow us to represent the set of requirements for both services and business processes.

To describe this relationship, we show a connection between a *DynamicCondition* and a *ModelingElement*. This connection reflects the *scope* of a quality requirement, i.e., the set of functional model elements it affects. It is easy to conclude that in our model, the scope of a quality requirement is a set of functional items (actors, operations, and events). This set is formed based on the set of the item conditions (depicted via the *ItemCondition* metaclass) belonging to this requirement.

Calling Contexts

Services can be called in different contexts and that *calling context* may impact the required or possible quality (Wada, Suzuki, & Oba, 2006). To reflect this fact, for each functional item condition a *Context* can be specified and a rule that applies to an item instance of that kind if the context is the specified one. For example, consider an event such as "the order arrives" then this way we can specify that another event must take place and can even specify alternative responses by distinguishing different contexts. Context belongs to a specific *ContextType* (such as "connection" or "load"), for example "dialup" and "broadband" contexts belongs to a context type "network connection". We can also have a *default* or *void context* for the cases when the specification indicates that the rule always applies to that item instance.

Among the rules, we distinguish *conditional* (if-then-else) rules, and *negative rules* (permitting to specify that events must not occur under certain circumstances). Actors have a portfolio of operations they are entitled to perform under specified circumstances. *item condition* allows to change that portfolio for specified contexts.

Static Conditions: Integrating the Quality Model

A static condition is assumed to be a composite of elementary conditions each of which has specified parts and each of which has a quality, i.e., can be required to take place or be prohibited. The specified parts of such condition include a *QualityMetric* (described below), a *Comparator* and the *ValueSet* that describes a threshold for the given requirement. For example, requirement S11 above employs the comparator *below* and the threshold value *2 sec*.

Table 4. Excerpt from quality model predesign glossary representing WSQM

Quality model		WSQM (E. Kim & Lee, 2005) adaptation for the order processing domain							
Quality category		Quality characteristic		Quality metric					
id#	name	id#	name	id#	name	definition	unit	order	scale
Q1	Business value								
Q2	Service mea-sure-ment	Q3	Performance	Q4	Response time	time taken to send a request and receive the response	sec	mini-mize	absolute
				Q5	Maximum throughput	max number of services to be processed for a unit time		maxi-mize	…
		Q6	Stability	Q7	Availability	the ratio of time period in which a service is ready		maxi-mize	ratio
				Q8	Accessibility	the ratio of acknowledge-ments to the requests		maxi-mize	ratio

Glossary Representation of the Quality Model and Software Metrics

Our metamodel allows for a flexible representation of the composite elements related to quality measurement. Every elementary condition can refer to a quality metric which is used by this condition. The treatment of this metric can vary; in our metamodel, we show a simplified representation. However, a more complicated fragment of the metamodel (such as the metamodel corresponding to the measurement ontology described in (Bertoa, Vallecillo, & Garc¡a, 2006)) can be integrated instead. A corresponding glossary fragment is shown on Table 4, which uses table nesting to represent relationships between quality characteristics and metrics.

Glossary Representation of Quality Requirements

Scope expressions and operation-type specifications are specified in a requirement scope glossary shown in Table 5. This glossary corresponds to

Table 5. Defining requirement scope expressions in a requirement scope glossary

id#	name	classification	scope		…	req. source
QS1	all operations involving orders or order items	expression	D3, Order	parameter		S9
			OR			
			D5, Order item	parameter		
QS2	authorize operation	enumeration	O7, authorize			S10
QS3	order items operation	enumeration	O3, order items			S11
QS4	all operations called by order department	expression	D6, Order department	calling		S12
QS5	all order processing service operations	Expression	D1, Order processing service	executing		S14
QS6	all operations performed if the order comes in	Condition	C1, order comes in			S23
QS7	all operations performed if the order is rejected	Condition	C6, order is rejected			S24

Table 6. Service and business process quality requirements in a quality requirements glossary

id#	quality characteristic	Scope	Context information		…	threshold	description	req. source
			context type	context				
C1	Q4, Response time	QS1, all operations involving orders or order items				0,5		S9
C2	Q4	QS2				0,3		S10
C3	Q4	QS3, order items operation	T1, connection	dialup		2		S11
			T2, load	<500 users				
			T1, connection	dialup		5		S11
			T2, load	>500 users				
			T1, connection	non-dialup		0,5		S11
C4	Q7, Availability	QS4, all operations called by order department					highest possible	S12
C5	Q7	QS3					second highest	S13
C6	Q9, Authentication	QS5, all order processing service operations						S14
C7	Q7	QS6				99		S23
C8	Q4	QS7				0,3		S24

the target for the metaassociation between *ItemCondition* and *ModelingElement* metaclasses in a metamodel (Figure 4). To represent the requirement scope, the glossary contains what we call *scope expressions*. The parameters of an *elementary scope expression* P(T,I) are an involved thing-type *T* and involvement type *I* for the operation-types in question. For example, for requirement S14 above (which refers to all operations of OPS the order processing service) a scope expression will accept *T = "Order processing service"* and *I = "provides for execution"* so the resulting requirement scope will contain all the operation-types provided for execution by the OPS. Elementary scope expressions can be chained together using logical operators to form *complex scope expression*.

Table 6 shows a fragment of a quality requirements glossary related to our example. C4 and C5 are imprecise constraints, C3 is a context dependency. C1, C2, and C6 represent refined requirements. It corresponds to both dynamic conditions and calling contexts in a metamodel.

Our metamodel allows us to easily extend the quality modeling notation to business process quality requirements. It is easy to see that the dynamic condition for the quality requirements (represented by its scope in glossaries) can refer, among others, to cooperation-types as a whole, as well as to their pre- and post-conditions. As a result, it becomes possible to capture the semantics of quality requirements applied to any point in a workflow: both cooperation-types and operation-types in general.

Quality requirements may influence each other (a fact reflected by the *Impact* metaclass on Figure 4). That association can be specified in a related glossary (*requirement impact glossary*). For brevity, we omit this glossary here.

PREDESIGN ADVANTAGES AND USAGE EXAMPLES

Reasons for choosing the predesign approach to model service requirements are the following:

1. The approach applies a simple semantic model, and a representation that is well-known to the intended audience: business stakeholders. A glossary (tabular representation) may be used like a check list (Galle et al., 2008) which is particularly useful during requirements elicitation. Tabular representations complement elicitation whenever a graphical model is not adequate or would lead - if used - to numerous topological revisions (i.e. elicitation phase).
2. The approach is user-centered, participative, and restricts itself to a small set of modeling concepts.
3. The relational model (minus queries) encoded form of requirements predesign provides a definition of key concepts which allows to interpret these concepts as well in terms of the application domain as well as software items. This justifies an easy way of creating a first system prototype by simulating the application domain concepts as functionality chunks.
4. The definitions above show that application domain experts do not have to treat their domain in terms of formal or even implementation concepts. This is important because most of them are not educated for this and thus would be overstrained. Moreover, they would be ruled out from validation in case of un-familiar formalisms.
5. The approach is integrated into a rich set of requirements analysis techniques utilizing natural language processing (Fliedl et al., 2002) and interactive simulation of the system under development (Kaschek et al., 2008)
6. A rich set of mapping techniques allows the predesign model to be mapped into other semantic models (design-time) such as UML or Semantic Web Services languages.
7. Using the NF^2 meta-metamodel allows for performing complex relational queries over the predesign model. We will discuss this below.

Table 7a. Tradeoff-supporting query

thing-type	involvement	constraint
		id#
D1, order processing service	executing	C1
		C3
		…
		C6

Relational and Dimensional Queries over Glossaries

We will look in more detail at one particular advantage of our relational predesign: *that all quality information is contained in glossaries, i.e., models of similar structure*. We think that generic queries can be worked out and issued against glossary sets that calculate the impact of any quality requirement on required functionality or quality characteristics. That would make it possible to estimate the effect of any requirements changes on selected quality characteristics or functionality.

Assume, for example, that we want to consider the effect of any requirements changes to OPS the “order processing service”. Then we need to know all the requirements that have OPS in their scope (see Table 7a). It is not difficult to design a query that would do the job. Something like “SELECT id# FROM RSG WHERE OPS IN scope” will essentially do if RSG is the requirements scope glossary in Table 4.

Relational predesign could simplify to make tradeoffs between different quality characteristics. Suppose that we want, with the given resources, at a time to ensure minimal response time and maximal availability. By issuing a query for each constraint against CIG the constraint impact glossary (not included in this paper) we can identify those operation-types that impact the constraint positively (towards the optimum) or negatively

Table 7b. Maintenance-supporting query

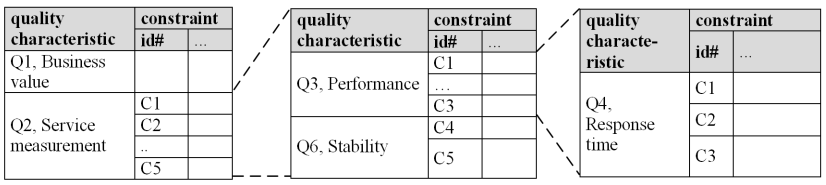
Context information		operation-type	
context type	**context**	**id#**	**name**
T1, connection	dialup	O3	order items
		…	

(away from the optimum). This allows us (by investigating the scopes of impacting requirements) at least to identify those operations the implementation of which aids at the same time meeting both requirements. We also can identify those operations that are counter-productive with respect to achieving the two requirements.

Relational predesign also can play a role in system maintenance as changed requirements could be related to those parts of programs that need to be changed. For example assume that we plan a system upgrade from using dialup lines to more recent communication media. We might then have to find the functionality affected by the requirements related to "dialup". The query corresponding to this request has to find the set of requirements referring to the "dialup" context and then identify the operation-types belonging to the scopes of these requirements. Resulting glossary can be seen on Table 7b. It is also possible to explore the query further using functionality-related part of the predesign model, e.g. finding all services providing these operations for execution or all the thing-types calling these services.

Interesting query possibilities can be also opened by exploring the obvious similarity between the metamodel for quality requirements (Table 4) and data schemas commonly used in dimensional data modeling (Kimball & Ross, 2002) such as a star or snowflake schema. Using the terminology of dimensional modeling, we can state that the requirements glossary can be seen in a way similar to a *fact table* while quality model glossary and, to less degree, scope and context (thing-type) glossaries can be treated similarly to *dimension tables* (quality, functionality and context dimensions, respectively). Exploiting this similarity further, we can state that many query and presentation techniques originated in dimensional modeling and OLAP communities can be exploited this way. For example, if we take a look at the Quality dimension we can see that its hierarchical organization allows implementing the "drill down" operation: for advancing from high-level quality characteristic down to lower-level ones narrowing the sets of requirements received along the way – up to pointing to the set of characteristics affected to the particular low-level metric (Figure 5). Aggregate functions like *count* of relevant requirements, their *average* impact, or the *smallest* established threshold can be calculated as well instead of receiving the lists of requirements. As an example, consider an analyst who would like to see the largest throughput required for all operations in a system in order to make sure that the set of requirements is compliant to the current hardware capabilities. To do this, it is possible to drill from "Service Measure-

Figure 5. Example of drilling down the quality dimension



quality characteristic	constraint	
	id#	**…**
Q1, Business value		
Q2, Service measurement	C1	
	C2	
	..	
	C5	

quality characteristic	constraint	
	id#	**…**
Q3, Performance	C1	
	…	
	C3	
Q6, Stability	C4	
	C5	

quality characte-ristic	constraint	
	id#	**…**
Q4, Response time	C1	
	C2	
	C3	

Table 8. Requirements grouped by quality scope, quality characteristic and context

quality scope	constraint			quality characteristic	constraint			context information		constraint	
	id#	...			id#	...		context type	context	id#	...
QS1	C1			Q4, Response time	C1			T1, connection	dialup	C3	
QS2	C2				C2				non-dialup	C3	
QS3	C3				C3			T2, load	<500 users	C3	
	C5			Q7, Availability	C4					…	
QS4	C4				C5			default		C5	
QS5	C6			Q9, Authentication	C6					C6	

ment" high-level characteristic down to the "maximum throughput" metric and to obtain the value of the aggregate function "MAX(threshold)".

Another dimension of drilling down can be the functionality. Here one can proceed from the coarser-grained functional artifacts (such as thing-types representing services) down to fine-grainer ones (such as particular service operations).

The "slice and dice" operation can be performed as well. In this operation, the information can be presented from several points of view on the user's request. For example, the quality requirements in a glossary can be grouped by affected functionality (starting from quality scope), quality characteristic and context (Table 8).

Similarities between Business Processes and Data-Centric Information Systems

Our approach allows us to highlight and try to exploit the analogon between our notion of business process (as PAIS backed by BP engine enacting NF^2 definitions of business processes) and data centric information systems that are facilitated by DBMS "enacting" table definitions in relational databases. We think this is a very new viewpoint that helps better understand BP quality.

RELATED WORK

Modeling Service Quality Requirements

Song (2007) discusses three challenges related to developing quality requirements for service-oriented applications: flexibility of service deployment, sharing services among stakeholders, and a more complex cost structure. (Wada et al., 2006) provides UML stereotypes for conceptualizing service, message exchange, message, connector and filter for modeling non-functional service attributes in a service-oriented application. The key elements of that approach, as listed above, correspond to the low-level (technical) aspects of service implementation, which are not well suited for end-user verification. That model was enhanced in (Wada, Suzuki, & Oba, 2007) with a feature modeling technique from (Czarnecki & Eisenecker, 2000) to represent the requirements. That technique allows a detailed treatment of interdependencies among quality characteristics that are treated as features of the service under development. Another metamodel for non-functional properties of services with their classification based on scales of measurement was proposed in (Hündling, 2005).

(Jingbai et al., 2008) provides an approach to modeling quality service requirements. A domain process model (DPM) is introduced that includes dynamic system behavior aspects with notions similar to operation-types and cooperation-types. Integration of quality-related information into DPM is explained. That paper deals with the non-functional context of the service invocation on the metamodel level. That has actually influenced our solution. The differences between our approaches are, however, quite significant. In particular, we have worked out the business process model much further. Also their approach does not include inter-dependencies between quality requirements and tasks. The invocation context, though similar in meaning, is simpler in structure and allows only textual representation.

Modeling Business Process Quality Requirements

Quality requirements to business processes are similar in organization to the traditional quality requirements. Among the modeling efforts (Pavlovsky & Zou, 2008) introduced the notion and graphical representation of so called quality constraint condition which can be applied to different steps of the process. Several papers use aspect-oriented techniques for expressing quality requirements and quality attributes of business processes. In (Wada et al., 2006) and (Wada et al., 2007) a method is specified for modeling non-functional business processes characteristics. (Wada, Suzuki, & Oba, 2008) uses the language of aspect-orientation for discussing crosscutting relationships. That approach, however, is not well suited for expressing relationships between quality and such process functionality that can be verified by business stakeholders. An aspect-oriented approach to introduce QoS into BPEL utilizing WS-Policy (“Web Services Policy Framework,” 2006) for QoS policy description is (Charfi, Khalaf, & Mukhi, 2007).

(Aburub et al., 2007) is devoted to establishing an approach to represent quality requirements to business processes; in particular, it proposes the corresponding quality model. The problems with this paper are connected with the fact that it equates without further discussion the quality of the process they have studied with the quality of the data that process obtains. This may be in line with the requirements of the related project. We think that was not really clear from that paper. For example, it is obviously an important process quality aspect beyond the data quality how much running one process instance costs and how long it takes to do so. These quality characteristics were ignored in the paper.

Service Quality for the Semantic Web Services Framework

Predesign modeling of service requirements is closely related to the Semantic Web Services approach. (Cappiello, Pernici, & Plebani, 2005) introduce a general semantic model that features the quality dimension (actually a metamodel for some quality model) and provide an integration with service representation describing the QoS. Possible ways of representing this model using OWL are discussed too. Another approach aimed at creating a QoS ontology suitable for integration in any web services description ontology is presented in (Tondello & Siqueira, 2008), this ontology is called QoS-MO.

A comprehensive comparison of OWL-S and WSMO is done in (Lara et al., 2005). It considers, among others, the possibility to incorporate QoS-related information into the related ontologies. Some specific approaches to achieve this goal are known. In particular, an approach to enhance WSMO with QoS ontology is introduced in (Toma, Foxvog, & Jaeger, 2006; Wang, Vitvar, Kerrigan, & Toma, 2006). A techniques for enhancing OWL-S with information based on UML Profile for QoS is discussed in (Celik & Elci, 2008; Kritikos & Plexousakis, 2007; Schröpfer, Schönherr, Offermann, & Ahrens, 2007). Defining quality-related characteristics through policies on the level of BPEL-defined processes is described in Baresi, Guinea, and Plebani (2007).

The predesign approach and Semantic Web Services are, in our view, complementary and can be integrated. That requires: (1) representing service ontologies using the predesign model in a way similar to domain ontologies (Kop, Mayr, & Zavinska, 2004); (2) Integrating quality-related information into these ontologies; and (3) mapping semantic service information into a specific service description language.

CONCLUSION

In this chapter, we have proposed an extension of the predesign model initially defined in Kop and Mayr (2002), Mayr and Kop (1998), and Shekhovtsov et al. (2008). By strictly employing NF^2-tables we progressed to *Relational Predesign*. That extension is for modeling quality requirements for services and business processes. We showed that it flexibly integrates specific or standard quality models for both services and business processes, separates functional and quality-related concerns and flexibly describes the requirements semantics through relationships between these concerns.

Predesign can contribute to improved requirements quality because it is integrated, as technology exists for mapping predesign models onto UML and other conceptual languages. It is open, as new elicitation techniques can be easily integrated into the predesign. It is participative, since our targeted audience easily can comprehend and manage its models. Predesign employs only few modeling concepts and thus is easy to learn. It is sound, as technology exists for analyzing predesign models. We consider predesign as stakeholder centered, because it empowers stakeholders to drive the requirements process.

REFERENCES

Abramowicz, W., Zyskowski, D., Suryn, W., & Hofman, R. (2007). SQuaRE based Web services quality model. In [IAENG.]. *Proceedings of IMECS*, *08*, 827–835.

Aburub, F., Odeh, M., & Beeson, I. (2007). Modelling non-functional requirements of business processes. *Information and Software Technology*, *49*(11), 1162–1171. doi:10.1016/j.infsof.2006.12.002

Al-Masri, E., & Mahmoud, Q. H. (2008). Toward quality-driven Web service discovery. *IT Professional*, *10*(3), 24–28. doi:10.1109/MITP.2008.59

Andreozzi, S., Montesi, D., & Moretti, R. (2003). Web services quality. In [IEEE Press.]. *Proceedings of CCCT*, *03*, 252–257.

Baresi, L., Guinea, S., & Plebani, P. (2007). Policies and aspects for the supervision of BPEL processes. In *Proceedings of CAiSE'07,* (LNCS 4495), (pp. 340-354). Springer.

Bertoa, M., Vallecillo, A., & Garcia, F. (2006). An ontology for software measurement. In Calero, C., Ruiz, F., & Piattini, M. (Eds.), *Ontologies for software engineering and software technology* (pp. 175–196). Springer. doi:10.1007/3-540-34518-3_6

Cappiello, C., Pernici, B., & Plebani, P. (2005). Quality-agnostic or quality-aware semantic service descriptions? In *Proceedings of W3C Workshop on Semantic Web Service Framework*.

Celik, D., & Elci, A. (2008). Semantic QoS model for extended IOPE matching and composition of Web services. In [IEEE CS Press.]. *Proceedings of COMPSAC*, *08*, 993–998.

Chaari, S., Badr, Y., Biennier, F., BenAmar, C., & Favrel, J. (2008). Framework for Web service selection based on non-functional properties. *International Journal of Web Services Practices*, *3*(1-2), 94–109.

Charfi, A., Khalaf, R., & Mukhi, N. (2007). QoS-aware Web service compositions using non-intrusive policy attachment to BPEL. In *Proceedings of ICSOC'07,* (LNCS 4749), (pp. 582-593). Springer.

Codd, E. F. (1979). Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems*, *4*(4), 397–434. doi:10.1145/320107.320109

Czarnecki, K., & Eisenecker, U. (2000). *Generative programming: Methods, tools and applications*. Addison-Wesley.

Dobson, G. (2004). *Quality of Service in Service-Oriented Architectures*. Dependability Infrastructure for Grid Services Project.

Dobson, G., & Sanchez-Macian, A. (2006). Towards unified QoS/SLA ontologies. In [IEEE CS Press.]. *Proceedings of SCW*, *06*, 169–174.

Dumas, M., Van der Aalst, W., & ter Hofstede, A. (2005). *Process-aware Information Systems*. Wiley-IEEE. doi:10.1002/0471741442

Evans, J., & Filsfils, C. (2007). *Deploying IP and MPLS QoS for multiservice networks: Theory and practice*. Morgan Kaufmann.

Fliedl, G., Kop, C., Mayerthaler, W., Mayr, H. C., & Winkler, C. (2002). The NIBA approach to quantity settings and conceptual predesign. In *Proceedings of NLDB'01*.

Galle, D., Kop, C., & Mayr, H. C. (2008). A uniform Web service description representation for different readers. In [IEEE CS Press.]. *Proceedings of ICDS*, *08*, 123–128.

Galster, M., & Bucherer, E. (2008). A taxonomy for identifying and specifying non-functional requirements in service-oriented development. In *Proceedings of 2008 IEEE Congress on Services - Part I,* (pp. 345-352). IEEE CS Press.

Guceglioglu, A. S., & Demirors, O. (2005). Using software quality characteristics to measure business process quality. In *Proceedings of BPM'05,* (LNCS 3649), (pp. 374-379). Springer.

Heitmeyer, C. L. (2007). Formal methods for specifying, validating, and verifying requirements. *Journal of Universal Computer Science*, *13*(5), 607–618.

Hündling, J. (2005). Modelling properties of services. In *Proceedings of 1st European Young Researchers Workshop on Service Oriented Computing*.

Janicki, R., Parnas, D. L., & Zucker, J. (1997). Tabular representations in relational documents. In Brink, C., Kahl, W., & Schmidt, G. (Eds.), *Relational methods in computer science*. Berlin: Springer.

Jingbai, T., Keqing, H., Chong, W., & Wei, L. (2008). A context awareness non-functional requirements metamodel based on domain ontology. In *Proceedings of IEEE International Workshop on Semantic Computing and Systems,* (pp. 1-7). IEEE CS Press.

Jureta, I.J., Herssens, C. & Faulkner, S. (2008). A comprehensive quality model for service-oriented systems. *Software Quality Journal.*

Kaschek, R., Kop, C., Shekhovtsov, V. A., & Mayr, H. C. (2008). Towards simulation-based quality requirements elicitation: A position paper. In *Proceedings of REFSQ 2008,* (LNCS 5025), (pp. 135-140). Springer.

Kassab, M., Ormandijeva, O., & Daneva, M. (2008). A traceability metamodel for change management of non-functional requirements. In *Proceedings of International Conference on Software Engineering Research, Management, and Applications,* (pp. 245-254). IEEE CS Press.

Kim, E., & Lee, Y. (2005). *Quality model for Web services 2.0*. OASIS.

Kim, H. M., Sengupta, A., & Evermann, J. (2007). MOQ: Web services ontologies for QoS and general quality evaluations. *International Journal of Metadata*. *Semantics and Ontologies*, *2*(3), 195–200. doi:10.1504/IJMSO.2007.017612

Kimball, R., & Ross, M. (2002). *The data warehouse toolkit: The complete guide to dimensional modeling* (2nd ed.). Wiley.

Kop, C., & Mayr, H. C. (2002). Mapping functional requirements: From natural language to conceptual schemata. In. *Proceedings of SEA*, *02*, 82–87.

Kop, C., Mayr, H. C., & Zavinska, T. (2004). Using KCPM for defining and integrating domain ontologies. In *Proceedings of Web Information Systems - WISE 2004 Workshops,* (LNCS 3307), (pp. 190-200). Springer.

Kritikos, K., & Plexousakis, D. (2007). OWL-Q for semantic QoS-based Web service description and discovery. In *Proceedings of SMRR'07.*

Lara, R., Polleres, A., Lausen, H., Roman, B., de Bruijn, J. & Fensel, D. (2005). *A conceptual comparison between WSMO and OWL-S.* (WSMO Deliverable 4.1).

Lee, Y., Bae, J., & Shin, S. (2005). Development of quality evaluation metrics for BPM (Business Process Management) system. In [IEEE CS Press.]. *Proceedings of ICIS*, *05*, 424–429.

Lee, Y., & Yeom, G. (2007). A research for Web service quality presentation methodology for SOA framework. In [IEEE CS Press.]. *Proceedings of ALPIT*, *07*, 434–439.

Liu, Y., Ngu, A. H. H., & Zeng, L. (2004). QoS computation and policing in dynamic Web service selection. In [ACM Press.]. *Proceedings of WWW*, *04*, 66–73.

Mayr, H. C. (2006). Conceptual requirements modeling–a contribution to XNP (eXtreme Non Programming). In *Proceedings of APCCM'06,* (CRPIT, Vol. 53). Australian Computer Society.

Mayr, H. C., & Kop, C. (1998). Conceptual predesign-bridging the gap between requirements and conceptual design. In [IEEE CS Press.]. *Proceedings of ICRE*, *98*, 90–100.

Mayr, H. C., Kop, C., & Esberger, D. (2007). Business process modeling and requirements modeling. In *Proceedings of ICDS'07,* (p. 8). IEEE CS Press.

O'Sullivan, J., Edmond, D., & ter Hofstede, A. (2002). What's in a service? Towards accurate description of non-functional service properties. *Distributed and Parallel Databases*, *12*, 117–133. doi:10.1023/A:1016547000822

Pastor, O. (2006). From extreme programming to extreme non-programming: Is it the right time for model transformation technologies? In *Proceedings of DEXA'06,* (LNCS 4080), (pp. 64-72). Springer.

Pavlovsky, C. J., & Zou, J. (2008). Non-functional requirements in business process modeling. In *Proceedings of APCCM'08 - Vol 79,* (pp. 103-112). Australian Computer Society.

Ran, S. (2003). A model for Web services discovery with QoS. *ACM SIGecom Exchanges*, *4*(1), 1–10. doi:10.1145/844357.844360

Schek, H., & Pistor, P. (1982). Data structures for an integrated database management and information retrieval system. In. *Proceedings of VLDB*, *82*, 197–207.

Schröpfer, C., Schönherr, M., Offermann, P., & Ahrens, M. (2007). A flexible approach to service management-related service description in SOAs. In *Emerging Web Services Technology, Part II.* (pp. 47-64). Basel: Birkhäuser.

Shekhovtsov, V. A., Kop, C., & Mayr, H. C. (2008). Capturing the semantics of quality requirements into an intermediate predesign model. In *Proceedings of SIGSAND-EUROPE'2008 Symposium,* (pp. 25-37). GI.

Song, X. (2007). Developing non-functional requirements for a service-oriented software platform. In [IEEE CS Press.]. *Proceedings of COMPSAC*, *07*, 495–496.

Toma, I., Foxvog, D., & Jaeger, M. C. (2006). Modeling QoS characteristics in WSMO. In *Proceedings of MW4SOC'06* (pp. 42–47). ACM Press. doi:10.1145/1169091.1169098

Tondello, G. F., & Siqueira, F. (2008). The QoS-MO ontology for semantic QoS modeling. In *Proceedings of 2008 ACM Symposium on Applied Computing,* (pp. 2336-2340). ACM Press.

Vanderfeesten, I., Cardoso, J., Mendling, J., Reijers, H. A., & van der Aalst, W. (2007). Quality metrics for business process models. In *2007 BPM and Workflow Handbook,* (pp. 179-190). Future Strategies Inc.

Wada, H., Suzuki, J., & Oba, K. (2006). Modeling non-functional aspects in Service Oriented Architecture. In *Proceedings of SCC'06,* (pp. 222-229). IEEE CS Press.

Wada, H., Suzuki, J., & Oba, K. (2007). A feature modeling support for non-functional constraints in Service Oriented Architecture. In *Proceedings of SCC'07,* (pp. 187-195). IEEE CS Press.

Wada, H., Suzuki, J., & Oba, K. (2008). Early aspects for non-functional properties in service oriented business processes. In *Proceedings of 2008 IEEE Congress on Services - Part I,* (pp. 231-238). IEEE CS Press.

Wang, X., Vitvar, T., Kerrigan, M., & Toma, I. (2006). A QoS-aware selection model for Semantic Web services. In *Proceedings of ICSOC'06,* (LNCS 4294), (pp. 390-401). Springer.

Web Services Policy Framework. (2006). *Policy outline.* Retrieved from http://www.w3.org/Submission/WS-Policy/

Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., & Chang, H. (2004). QoS-aware middleware for Web services composition. *IEEE Transactions on Software Engineering*, *30*(5), 311–327. doi:10.1109/TSE.2004.11

Zhou, J., Niemelä, E., & Savolainen, P. (2007). An integrated QoS-aware service development and management framework. In *Proceedings of WICSA'07*. IEEE CS Press.