Diploma Thesis

April 21, 2006

A Quality-of-Service Model for Loosely-Coupled Peer-to-Peer Workflows

Matthias Taugwalder

of Zermatt, Switzerland (00-921-866)

supervised by

Prof. Dr. Harald Gall

Dr. Gerald Reif

University of Zurich

Department of Informatics

s.e.a.l.

software evolution & architecture lab

Diploma Thesis

A Quality-of-Service Model for Loosely-Coupled Peer-to-Peer Workflows

Matthias Taugwalder

University of Zurich
Department of Informatics

s.e.a.l.
software evolution & architecture lab

Diploma Thesis

Author: Matthias Taugwalder, matthias.taugwalder@bluewin.ch

Project period: October 26, 2005 - April 21, 2006

Software Evolution & Architecture Lab
Department of Informatics, University of Zurich

Acknowledgements

I would like to thank my supervising assistant Gerald Reif for his valuable input, the extensive proofreading and the freedom I had while writing this thesis. Further, I thank Prof. Harald Gall for giving me the opportunity of writing this thesis.

Many thanks also to my parents Agathe and German for their support during the last years.

Abstract

The Internet and the development of new technologies such as Peer-to-Peer networks have changed the way how companies do business together. Today anyone can provide all sorts of services to the global community. This entails that service consumers can choose from a mass of different service providers. But each service provider offers the service to different conditions. Also the service requester may have specific requirements of the service. That is where the quality of a service becomes important.

Quality of Service (QoS) is a framework for specifying service attributes which helps the service requester to choose between different service providers. In the traditional sense this QoS was limited to technical aspects such as network speed and failure rate. The usage of QoS in other domains such as Web services and Workflow systems introduced additional QoS criteria. Furthermore, different users may have different needs: While for system administrators the technical aspects of a service are important, the end-user may have additional requests and also a totally different perception of the service and its quality.

This diploma thesis describes an open and extensible framework for specifying QoS criteria and service attributes in the domain of Peer-to-Peer systems. The thesis introduces the terminology in the domain and discusses existing QoS approaches. It further addresses necessary considerations of the specification process, which help to produce meaningful and useful QoS measures and results. Finally this thesis presents a proof-of-concept implementation of the proposed extensible QoS model that is based on Java and RDF (Resource Description Framework) language.

Zusammenfassung

Das Internet und die Entwicklung von neuen Technologien wie Peer-to-Peer Netzwerken haben die Zusammenarbeit zwischen Unternehmungen verändert. Jeder kann heutzutage alle Arten von Dienstleistungen global anbieten. Dies bringt mit sich, dass die Servicekonsumenten aus einer Vielzahl von verschiedenen Dienstleitungsanbietern auswählen können. Jedoch bietet jeder Anbieter seine Dienstleistung zu unterschiedlichen Konditionen an. Zudem kann der Konsument spezifische Anforderungen an den Service haben. An dieser Stelle wird die Qualität einer Dienstleistung wichtig.

"Quality of Service" (kurz: QoS) bietet einen Rahmen um die verschiedenen Eigenschaften einer Dienstleistung zu beschreiben und hilft damit dem Konsumenten der Dienstleistung bei der Auswahl aus verschiedenen Anbietern. Im traditionellen Sinn beschränkt sich QoS auf technische Eigenschaften wie Netzwerkgeschwindigkeit oder Fehlerrate. Die Benutzung in anderen Gebieten wie Web Services oder in Workflow Systemen hat zu zusätzlichen QoS Kriterien geführt. Darüber hinaus können verschiedene Nutzer unterschiedliche Anforderungen haben: Während für Systemadministratoren die technischen Eigenschaften einer Dienstleistung entscheidend sind, können die Endnutzer zusätzliche Anforderungen haben und einen Service und die damit verbundene Qualität völlig unterschiedlich wahrnehmen.

Diese Diplomarbeit beschreibt ein offenes und erweiterbares Rahmenwerk zur Beschreibung von QoS Kriterien und Service-Attributen im Gebiet von Peer-to-Peer Systemen. Schrittweise werden die Fachbegriffe dieses Einsatzgebiets sowie bestehende QoS Ansätze vorgestellt. Die Arbeit befasst sich desweiteren mit Überlegungen zum Spezifikationsprozess, die dabei helfen aussagekräftige und brauchbare QoS Messwerte und Resultate zu erzeugen. Zu guter Letzt zeigt die Implementierung des vorgestellten Ansatzes, wie ein solches erweiterbares QoS Modell mittels Java und RDF (Resource Description Framework) realisiert werden kann.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

1.1 Motivation

The Internet and the development of new technologies such as Peer-to-Peer networks have changed the way of doing business. Peer-to-Peer networks offer new powerful ways for providing services. Anyone can provide a service to the global community. Service requesters can choose from a wide variety of service providers. Hence we need extensible ways for describing these service interactions and the quality of these services.

Therefore, to represent today's services rich and accurate means for describing service capabilities and attributes are required. That applies also to describing quality of service criteria for these services. Most of today's existing approaches concentrate on the technical aspects and not on the end-user perspective. What is needed, is a extensible model which is motivated by the every-day services that surround us.

1.2 Problem Description

The traditional concept of quality of service is insufficient for describing any service.

On the one hand, the existing approaches are too concentrated on the technical aspects. The existing definition of quality of service has to be reconsidered and expanded to involve all service attributes which are important to the service requester. There are several attempts in the domains of Web Services or Workflow systems which propose such models that focus on the end-user and comprehend the end-user's requirements and needs.

On the other hand, today's services are multifaceted and are hard to describe. The existing approaches for specifying the quality of a particular service are limited and not extensible. Business-specific properties cannot be expressed using these models. An extensible, open framework is required that allows to specify the quality of service parameters for any kind of services, including non-digital ones.

1.3 Structure of this Thesis

This thesis focusses on the aspects and problems, that have to be addressed while specifying a wide and extensible quality of service (QoS) model. The following chapters approach this problem step-wise.

Chapter 2 introduces the reader to the application area of loosely-coupled Peer-to-Peer networks and the domain of QoS. It deals with the theoretical background such as the fundamental definitions of a service and QoS. Furthermore, it presents non-functional properties as framework for classifying service constraints. Non-functional properties involve attributes a service can have.

In Chapter 3 we discuss the specification process itself. Having a powerful model is not enough, we need a careful process to find the right measures to produce meaningful results. On the one hand, the results have to be operationally meaningful to the end-users. On the other hand, the measures should be possible to produce with minimal effort.

Chapter 4 shows a generic QoS model which will be used to classify QoS attributes from the perspective of the end-user. After this, it presents and compares the existing approaches of quality of service models in several domains such as IP networks, Web services or Workflow systems. Out of these different models the essential features of a model are specified, which will be used to formalize our QoS model.

The resulting QoS model can be found in Chapter 5 as a basic QoS model for workflow services in a Peer-to-Peer environment. The proposed model is extensible and can be extended to the requirements of any service. Using three fictive scenarios its extensibility and expressiveness are shown.

The MOTION Peer-to-Peer plattform builds the basic system for the implementation. Chapter 6 describes these underlaying system components and the implementation of our QoS model.

Finally, in Chapter 7 we will draw a conclusion of the chosen approach and address some potential areas of future work.

Chapter 2

Theoretical Background

The popularity of Internet technologies led to the development of complex software systems, consisting of components which are distributed over the network. To keep such system maintainable, each component is specialized on a specific task or service. Such an architecture enables that services can be offered by different parties. To be able to choose the appropriate service, that matches the current requirements, the quality of a service is an important criteria.

Loosely-coupled Peer-to-Peer networks solve the problem of permanent network connectivity and limited availability between the single peers. The clients gather all required information if they are connected to the network and store that information in a local knowledge database. This database then provides all the necessary information if a peer is disconnected from the network.

In this chapter, we first deal with definitions of commonly used terms as "service" or "quality" (see Section 2.1). After that, we will try in Section 2.2 to cover and understand what service aspects are important to the quality of a service. Non-functional properties give an overview of different service aspects and provide a framework for classifying such service constraints.

2.1 Definitions

The following definitions are presented in [GJS03] which concentrates mainly on the quality of service terminology in IP networks.

2.1.1 Service

According to the International Telecommunications Union (ITU) a service in an IP environment is defined as follows:

> "A service provided by the service plane to an end user (e.g., a host [end system] or a network element) and which utilizes the IP transfer capabilities and associated control and management functions, for delivery of the user information specified by the service level agreements." [ITU01]

Most definitions are focused on the technical aspects of services. These aspects can be specified in a formal way in a Service Level Agreement (SLA) and can easily be verified.

In economics the definitions are mainly centered around the end-user: Services are described as non-material equivalents of goods, the customer takes part in the process of service provision.

On the other hand, in the domain of Web services a "Web service" is defined as:

> "A Web service is a software system designed to support interoperable machine-to-machine interaction over a network." [BHM+04]

A Web service has an interface described in a machine-processable format. Other systems interact with the Web service in a manner prescribed by its description using messages. [BHM+04].

2.1.2 Quality

The term quality is in the ISO 8402 standard document defined as:

> "The totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs." [ISO94]

This term is very broad and can be therefore interpreted in different ways: Different users have different needs and for each user different criteria is important. Also the evaluation if the service satisfies the user's expectations depends on various criteria. For example engineers would decide on technical aspects while end users decide on their personal impression and in comparison to their expectations.

2.1.3 Service Level Agreement (SLA)

In business the level of quality which providers offer and users expect is formalized using a service level agreement. So they can be considered as binding contracts that are agreed between a service provider and service requester. Usually there are penalties imposed in case of non-compliance. [OEH02]

The ITU definition of a service level agreement is:

> "A negotiated agreement between a customer and the service provider on levels of service characteristics and the associated set of metrics. The content of SLA varies depending on the service offering and includes the attributes required for the negotiated agreement." [ITU01]

According to the ITU definition, a SLA may be in form of a document containing names of the parties signing the contract. It should be composed of service level objectives, service monitoring components, and financial compensation components. Service level objectives contain QoS parameters or the class of the service provided, service availability and reliability, authentication issues, and so on. Service monitoring specifies the way of measuring service quality and other parameters used to assess whether the service meets the requirements which are specified in the SLA. The financial component may include billing options, penalties for breaking the contract, and so forth. [ITU01]

The Internet Engineering Task Force (IETF) defines a SLA in a similar way as:

> "A service contract between a customer and a service provider that specifies the forwarding service a customer should receive." [Cra98]

Furthermore, a SLA should be expressed in a way that the customer can understand. It includes a description of the basic features of the service and a list of unambiguous quality criteria which is used to assess if the service meets the stated requirements. The responsibilities of both parties - the service provider and also the customer - are defined in this document.

Summarizing, SLA is a broad term encompassing technical features and parameters of the service, as well as legal and charging aspects. SLA parameters and attributes define IP-based services.

2.1.4 Class of Service

Class of service is described as set of characteristics available with a specific service. The IETF describes it as "The definitions of the semantics and parameters of a specific type of QoS" [Cra98] Services belonging to the same class are described by the same set of parameters which can have qualitative or quantitative values.

In the domain of IP networks the class of service is used to ensure different levels of service provision. For example the IntServ architecture [BCS94] includes three classes: guaranted, controlled load and best effort.

2.2 Non-Functional Properties

As shown in [Tom06], three different aspects have to be considered when talking about service descriptions: (1) *functional*, (2) *behaviour*, and (3) *non-functional* properties. The *functional description* contains the formal specification of what exactly the service can do. The *behavior description* contains the formal specification of how the functionality of the service can be achieved. Finally, the *non-functional descriptions* capture constraints over the previous two [Chu91]. For example, in case of a train booking service, invoking its functionality (booking a train ticket) might be constrained by using a secure connection (security as non-functional property) or by actually performing the invocation of the services in a certain point in time (temporal availability as non-functional property).

Among these three, the first two aspects of service description are the most investigated aspects. Although non-functional properties did not capture a very broad attention, one has to recognize the big importance of describing them. This is due to their high relevance for all service related tasks.

The non-functional properties presented in [OEH02] are based on reviews of commercial services. Those constraints are exhibited over the functionality of the service. These non-functional properties of services include temporal and spacial availability, channels, charging styles, settlement models, settlement contracts, payment, service quality, security, trust, and ownership.

[OEH02] gives also an overview of existing solutions and approaches for each property. Further references to related work can be found in that paper.

2.2.1 Availability

Availability refers to temporal (i.e. when) and spatial (e.g. where) constraints of a service. It is a complex constraint, there are services on the move (e.g. taxis, trains), services where an implicit understanding of the advertised availability is needed (e.g. to attend a theatre you need to be in the lobby at the right time to take your seat), or others where there is a suspension and resumption (e.g. memberships).

Temporal and Spatial Issues

Service requesters may need to be aware of more than just the availability of service request and delivery times. To enable accurate scheduling of multiple services, the requester may be specially interested in the duration of the service or the approximate completion time. These may be required when performing service discovery, advertising, composition, and when determining service quality.

Temporal and Spatial Representation

Temporal representations need to support various granularities or alternatively represent time as relationship (e.g. service X begins after service Y). There are different approaches for representing these granularities. Spatial representations are used to describe topologies, orientation, shape, size, and distance.

2.2.2 Channels

We consider service interactions to occur using a channel. When describing a channel we need to take into considerations its endpoints, the information being transmitted, and the interaction pattern that occurs over the channel. Service description initiatives can be categorized by those, that describe service functionality or non-functional properties, and those that describe interactions with services.

2.2.3 Charging Styles

The styles presented here describe the charging technique applied by a service provider for the use of its service. Three styles are identified:

- per service request or delivery (e.g. fixed price local telephone call)
- by unit of measure and granularity (e.g. length, volume, weight, area or time)
- on a percentage or ratio basis of some aspect of the service (e.g. by commission)

Service providers may use an aggregation of these service styles. Sometimes the charge for a service is also redirected to another entity.

2.2.4 Settlement

Settlement is a process that reflects the mutual obligations of the provider and requester, with payment usually being an obligation of the requester, and service delivery being that of the provider. The settlement process is normally laid down by the provider, and is included as part of their business model. Sometimes the service process (and its sequence) is defined by the service environment.

Settlement Models

Packaging of obligations into a defined process results in a settlement model, which reflects the ordering and relationship between each party's obligations. None of these settlement models result in a transfer of ownership.

- Transactional: It can simply described as delivery for payment. It can include an one-off delivery or multiple deliveries. The later implies a longer term relationship.

- Rental: This model is the familiar concept of being "on loan". Explicit and spatial constraints may be imposed by the service provider (e.g. the rented good has to be returned by 2 pm tomorrow). Depending on the service it may include a short-term relationship (e.g. holiday unit) or long-term relationship (e.g. local video store membership).

Special forms of the transactional model are:

- Subscription: Implies a long-term relationship.

- Metered: It is almost identical to the basic transactional model, tracks consumption of the service except that the relationship may also impose restrictions making it difficult to change to another service provider.

- Facilitated: The provider acts as a conduit or facilitator to another service provider (e.g. a broker).

- Escrow: This is used when there is an identified trust issue, and where parties lodge their obligation with the escrow organisation.

- Swap: Parties agree that the services being traded are of equal value, and no payment is involved.

Multi-party settlements vary in the degree of binding between the parties involved. A tightly bound third-party might include a credit card provider (e.g. a bank) and a loosely bound third-party might include a company that provides software used during service provision.

Settlement Contracts

Terms and conditions which may be attached to a service are formalized in a contract and govern the responsibilities of all parties involved. Types of settlement contracts in an offline environment include

1. a Bill of Lading, which defines details of transportation (e.g. who, what, where) and what happens if something should go wrong.

2. a Promissory Note, which outlines the terms and conditions of a loan (e.g. required repayments, interest rate, and policies surrounding the loan).

Both parties must be agreeable with respect to the contract before it is invoked.

Recourse is available in some cases to either of both parties. In cases where obligations of either party are not realised there may be some level of re-negotiation performed.

2.2.5 Payment Obligations

Payment obligations may be required at any stage in the service invocation, provision, and execution process. These obligations are usually outlined to the service requester as part of the negotiation process. Service providers or their surrounding environment determine a valid set of payment instruments that are used to fulfill this obligation of the service requester. Further information about payment models can be found at [PASW97].

2.2.6 Service Quality

Service quality is a measure of the difference between expected and actual service provision. From the viewpoint of the requester, it measures the competence of the provider to deliver a service.

SERVQUAL [PZB88] is a way to measure these customer perceptions of the perceived service quality along five dimensions:

- Reliability: Dependability and accuracy of the service.
- Responsiveness: Promptness and willingness of the staff to assist.
- Assurance: Attributes of staff as knowledge and courtesy that convey trust and confidence to the user.
- Empathy: Level of caring and personalized attention provided to the requester.
- Tangibles: Concrete and physical aspects of the service, such as cleanliness.

Service providers may commit a certain level of quality. This commitment is sometimes formalized using Service Level Agreements (see Chapter 2.1.3).

2.2.7 Security and Trust

Security and trust are basic properties for electronic services.

Security

Security is increasingly being seen as a mandatory component for facilitating electronic commerce. It alleviates concerns relating to identity, privacy, alteration, and repudiation of information transferred between parties. Usually we think about ”on-the-wire” security that pertains to the request and delivery channels of a service, especially when the payment obligation of the service requester is being finalised. Common approaches for Security Protocols involve the implementation of a Public Key Infrastructure (PKI).

[OEH02] proposes that the individual aspects of service descriptions should be secured. For example used for distinct service descriptions for retail and wholesale clients, which will have different pricing and payment conditions. Alternatively, multiple advertisements could be generated by a service provider with access controls applied based on the type of requester accessing the information.

In the context of sub-services security becomes a more complex property. The following questions arise:

1. Should the client authenticate to all of the sub-services?

2. How can a service be stopped from being composed within another service?

3. What are the implications for a service if some sub-services require security and others do not?

4. What happens when sub-services have differing policies?

5. What constitutes an infringement of a security promise? How are infringements managed (e.g. penalty payment or removal from a composition)?

Trust

Trust can be defined as a reinforcing attribute that balances perceived risk, cost, and benefit [Mar94]. The same concerns are present in the service provision process. Trust can be mutual (i.e. both parties do not trust each other) or exclusive (e.g. the service provider trust the service requester but the service requester does not trust the provider).

Reputation mechanisms are an attempt to embody trust. The implementation of such mechanisms may be useful but concepts from non-electronic service provision may prove useful. People tend to be satisfied that when acting within a group they will be able to increasingly trust a service provider.

The following questions arise with respect to trust in service provision.

1. How do you represent trust of service providers or services requesters?

2. In a decentralised system how is knowledge relating to trust distributed, particularly changes to the perception of trust for a party? How do you trust a composition of services? Can an external party validate a service and/or provide a level of reputation based on previous interactions?

3. What are the implications or penalties for parties that are distrustful?

4. Does access to the past performance of a service provider reduce the perceived risk of the service requester?

2.2.8 Ownership and Rights

Provision of goods usually results in a change of ownership from the service provider to the service requester. Services do not involve a transfer of ownership. However, service requesters do have a limited set of rights that are associated with a service. These rights provide a degree of control over the request and consumption of the service. The rights available to service requesters are the following:

- The right to comprehend: service requesters should be able to question the provider with the intention of better understanding the service.

- The right to retract: once an advertised service offer has be refined in a service contract, via negotiation between service provider and service requester, the service requester can choose not to request an instance of that service. He maintains the right to request the service from another service provider.

- The right of premature termination: requesters may have the ability to prematurely terminate a service. The service provider may continue provision of the service and may choose to apply some form of penalty for partial consumption.
- The right of suspension: interrupting the delivery and therefore the consumption of a service can act as a useful method for extending the service provision process (e.g. a milkman who does not deliver while you are on holidays).
- The right to resumption: continues the delivery and consumption of a previously suspended service.

Chapter 3

Specification Process

One key issue for analyzing the QoS of a specific service is to find the right measures. On the one hand, they have to be defined in ways that are operationally meaningful and useful to the end-users who are using the service. On the other hand, the measurement and analysis should be possible to achieve with a minimum investment in time and money.

The following sections are based on [Har01] which concentrates on the measurement and evaluation of telecommunications quality of service. But the suggested analysis process can easily be extended and adapted to all kinds of services.

The analysis is a process whose result is to produce specific answers to specific questions. The analyst should address before specifying any measures the following questions:

- Audience: Which end-users will utilize the results of the analysis process?
- Utility: What kinds of decisions are to be facilitated? How must measurements be evaluated to produce information that can be used of those decisions?
- Concerns: What are the questions that those users are likely to want to have answered?
- Objectives: What are the courses of action that will be decided or determined by appeal to the results of the analysis?

3.1 Measurement and Evaluation

3.1.1 Classes of Decision-Makers

For telecommunications services there are at least five distinct classes of decision-makers who might be responsible for decisions. They can have a totally different user perception of QoS and the evaluation of measures needed to make the results useful is in each case different.

Service users

These are the actual users of the service. Experienced difficulties will affect at the end whether the perceived QoS is sufficient, thereby producing a subjective assessment of perceived quality. If the quality is unsatisfactory the user will not continue to use the service. These users are the ultimate decision-makers, their concerns are the principal focus of QoS measurement.

User Representatives

The end-users usually represent themselves in activities as choosing among competing providers of the chosen services and negotiating prices. These decision-makers are responsible for choosing, acquiring and maintaining services for a large body of users. Their primary interest are the user satisfaction with the selected services and therefore the perceived quality of service. Since their role is also one of assuring their management of economy of services, their perspective on QoS will be one of trying to assess cost-benefit trade-offs and they will be much more concerned with questions of billing and customer support.

Service Provider Sales and Marketing Personnel

The sales and marketing personnel are not necessarily decision-makers, but must respond to concerns with QoS raised by the end-users and the user representatives. The analysis must also show how quality of services they sell compares with that of competing services offered by other providers.

Service Operations and Maintenance Personnel

The service provider's operations and maintenance personnel are responsible for monitoring the day-to-day performance to assure that QoS is maintained at acceptable levels. Their focus is necessarily on intrinsic quality of service and their principal questions will be ones of the relationship between measures of intrinsic and perceived QoS. Analysis of perceived QoS has furthermore to produce derived indicators of specific conditions that must be corrected to avoid deleterious effects on the service user's assessment of perceived quality.

System Architects and Engineers

That are the part of the service provider's personnel who must make the decisions as to the technology to be employed in implementing the specific service and the way various assets are to be configured. Like operations and maintenance personnel, they are concerned with intrinsic quality and are responsible to for deciding the characteristics of the infrastructure and the allocation of resources that will achieve intrinsic quality adequate to assure a high likelihood that perceived quality will be acceptable. To do this, they must have hard and fast requirements that can be used as basis of system design and configuration. Notions of subjectivity and perception must be totally factored out of the equations and the fuzzy indicators that might be used for operations and maintenance management must be replaced by criteria for acceptability of variations of intrinsic quality that are technical, concrete, specific and completely unambiguous.

3.1.2 Concerns

If measures and quantifiers describe the what and how of measurement, then concerns explain the why. The term "concern" is used here as the rubric for an uncertainty that must be addressed in the evaluation of measurements. To make them concrete, such concerns will usually be described as a set of questions posed as to the likelihood of occurrence of undesirable events or conditions. The description of the concern nearly always defines the attribute to be measured so that in most cases we can use the same name for the measure and the concern, thereby making this association explicit.

3.1.3 Objectives

Finally, if concerns explain the reason for conducting an analysis, the objective(s) characterizes its envisioned utility to the intended audience. The term ”objective” does not refer to what the analyst is to accomplish, or what the analysis of QoS is to show. Rather, it is a description of the decisions to be made that generated the concerns to be addressed in the first place. Such objectives will then be described by completing the sentence: ”The results of this analysis will be used in deciding/determining where ... by ...”.

Just a formal description of concerns serves as a guide to selection of measures that will ensure an analysis of quality of service that is effective for its intended purpose. Selecting quantifiers of those measures based on a clear understanding of the objectives of the analysis will lead to the selection of the most cost-effective quantifiers for the defined measures. So before deciding what data is to be accumulated, there are two questions that must be answered:

- Who is likely the audience (decision-makers)?
- What are the objectives?

3.2 Analysis Process

Like the analysis, the process by which analysis comprising measurement and evaluation are conducted can be thought of as comprising multiple phases. The phases in this case are:

- Formulation, during which the audience, decisions supported, etc. are clarified and used as the basis for determining and specifying measurement requirements.
- Data handling, during which the data elements needed to quantify each measure are acquired, organized and manipulated.
- Evaluation, during which values of the measures are calculated and interpreted as necessary to address the specific concerns of the intended audience.

3.2.1 Formulation

In [Har01] a formal process is suggested that should be followed in structuring any analytical effort to assure that the end results will be operationally meaningful, useful to decision-makers, and achieved with a minimum investment in time and money. The principal steps are described in Figure 3.1, which displays the relationships among the six principal steps and the structure of a decision loop for selecting quantifiers.

The six steps are as follows.

Identify the Audience

As suggested in Figure 3.1 the first step in formulating any analytical effort is to determine the intended users of its results.

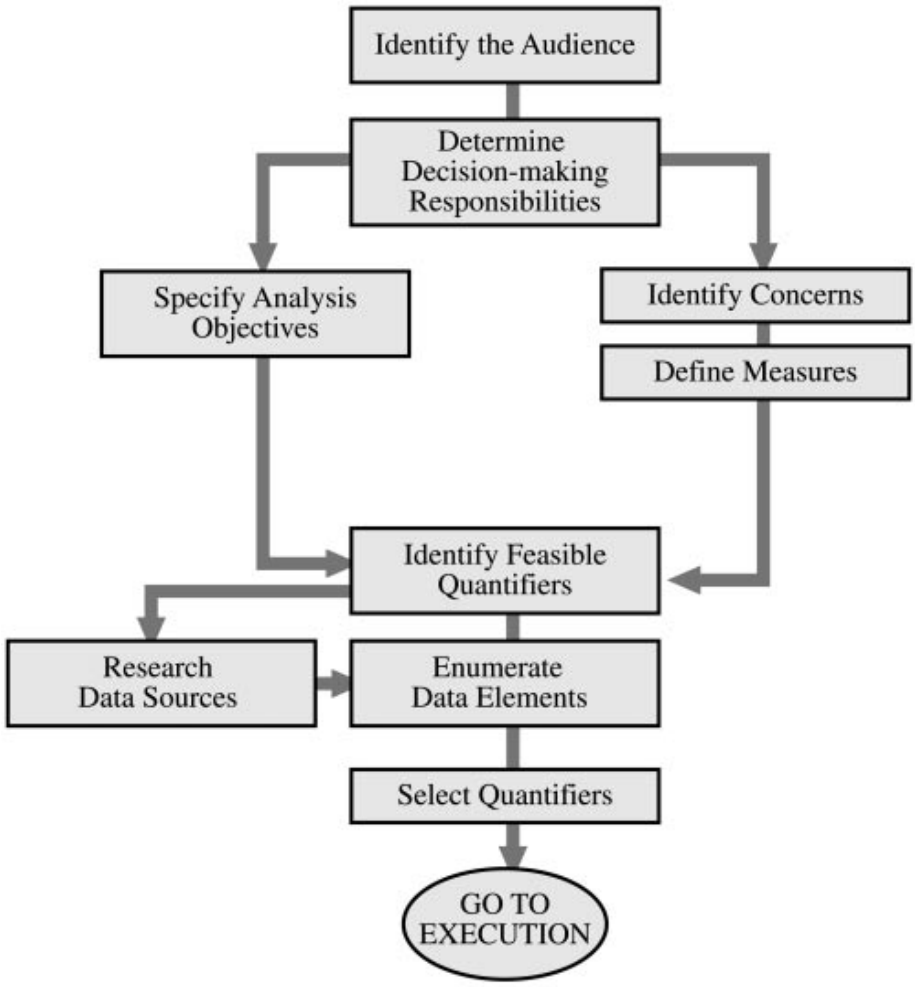

Figure 3.1: Process for Formulating an Analytical Effort [Har01]

Determine Decision-Making Responsibilities

Once the target audience is identified, the next step in the structured approach recommended is a conscious determination of the decisions or general kinds of decisions that will be facilitated by its results. As suggested earlier, those decisions will be some course of action with respect to the service, such as its purchase, continued use, marketing, operation and maintenance, or design.

For example, the basic user decision with respect to QoS is whether to keep the current service or shift to another. The alternative may simply be the same kind of service offered by a competing provider, or a new kind of service for meeting old requirements. The basic question to be made is always the same: Should I stick with the service I have or switch to something different?

Specify Analysis Objectives

A third step in the process, but not necessarily the third in order, is to review the decision-making responsibilities of the audience to formulate specific analysis objectives.

In our example before, the analysis of QoS should support the decision to buy or keep a particular service. The results of the analysis should enable users to make objective decision if a service that has not been experienced is satisfactory or not.

Identify Concerns

Having identified the decision-makers comprising the target audience and the decision that will be supported, it is also necessary to articulate the specific uncertainties that are likely to impede decision-making. Those uncertainties have been defined before as concerns, usually expressed in a form that can be readily understood by almost everyone.

Adopting only the measures of QoS targeted for a special user group, such as technically knowledgeable persons might be dangerous. The results of the analysis would not be likely convincing or helpful for the users' purposes of deciding what service to buy and how long to keep it. The reason is that users seldom buy, and frequently do not even understand, technology. Their perceptions of the quality of a service are instead based on how well that service meets their expectations and satisfies their needs when they use it. Thus, if the users cannot readily tell from an analysis based on technical measures what to expect from day-to-day use of the service, the results of the analysis will simply replace one set of uncertainties by another one, that are even harder to resolve.

Define Measures

As suggested in Figure 3.1, the definition of measures to be used in any analysis should be deferred until the relevant concerns of the intended audience have been identified. The time invested in the orderly formulation of the analysis will be amply rewarded by the ease with which useful, meaningful measures can be defined at this step. This effort should be so intuitive and natural that the attributes of the service to be measured will probably be identified in the description of the concerns, and the name of that attribute can readily be applied both to the concern and the measure without ambiguity.

Select Quantifiers

Once the measures have been defined and the analysis objectives have been clarified, it is then relatively easy to select the most cost-effective quantifier for each measure. The iterative steps in Figure 3.1 can be described as follows:

1. Identify feasible quantifiers: The objectives specified for the analysis will determine for each measure the possible acceptable forms or modes of quantifiers, determining, for example: whether a precise value is needed or an indicator will suffice; whether it will be more efficient to quantify the measure directly, or by estimating it as a function of sub measures, to ensure the ability to related observed values of the measure to measures of contributing factors that must be distinguished, etc.

2. Then, to assess the cost-effectiveness of each of the feasible quantifiers, each member of a set of acceptable quantifiers for a measure is considered in turn to: *Enumerate data elements* required to assign a value to the measure in accordance with the definition of the quantifier; and *Research data sources* to determine the ease with which the necessary data elements can be acquired.

3. On basis of the assessments in step (2), the quantifier for each measure is selected as the one among the feasible quantifiers for which the data elements can be most easily or most quickly acquired, depending on whether speed or ease of production of the analysis is the greater concern for the intended audience.

3.2.2 Data Handling

The first step in the analysis process produces a list of the data elements that are needed for the analysis. In the second phase of the process all those data will be acquired, organized and manipulated.

Data Acquisition

Enough of the right data has to be aggregated to support meaningful analysis and interpretation of the values selected for the analysis. We have defined yet *what* data sets are to be created and *where* the data comes from. Now has to be fixed *how much* data will be enough. This depends from how easy the data can be collected. If it comes from automatic sensors we will probably need all of that. On the opposite site, surveys or labor intensive observation processes have high data acquiration costs. In this case the question is, what amount of data will suffice the purposes of the analysis.

Data Organisation

Once questions of what data elements and sample sizes have been answered, the next major step is to begin to acquire the data and organize the samples into coherent databases where it can easily be retrieved from.

There are several ways to do this. [Har01] advises against using a DBMS (DataBase Management System) for storing the data. The data samples should be saved in raw mode if possible and queries are run over this data. Another approach is presented as using APL (A Programming Language), a programming language which first described in [Ive62] in 1962, implemented in 1966 and is still one of the fastest programming languages for rapid applications development.

Data Manipulation

Computer accessible databases remove the barriers to transforming the data to facilitate its understanding. The tools for such examination of data provide, for example:

- Visualization aids, which transform data into graphical displays, such as scatter diagrams, histograms, or time series charts.
- Calculation of distribution statistics, which transforms the data into numbers that describe how the data is distributed.
- Data fitting, comprising utilities to produce the best fit of data sets to a ”smooth” mathematical function. Such capabilities include for example linear regression.
- Data filtering, which is a process by which entire data sets are transformed by eliminating suspect, clearly erroneous, or useless elements. The objective is to ”clean up” data sets in order to make sure that all values are free of errors in data acquisition and represent what they are supposed to represent.

3.2.3 Evaluation

What was derived from the first two phases will now be interpreted to produce answers to specific questions posed by the decision makers. The results of this process will address questions related to the incidence and occurrence of undesirable conditions, outcomes or events by describing a likelihood of their occurrence. These undesirable conditions may be described in:

- Subjective, qualitative terms, such as: ”How often will we experience outages that will severely inconvenience our customers?”.
- Equivalent expressions with concrete examples, such as: ”How often will we experience outages of an hour and more?”.

Chapter 4

Quality of Service

Before specifying our quality of service model for loosely-coupled Peer-to-Peer workflows, we first discuss QoS definitions in the related work. After the discussion of the QoS definitions in different domains we introduce a categorization of QoS models. This categorization will be the basis for the definition of our QoS model in Chapter 5.

4.1 A Generic Quality of Service Model

As described in [Har01] many possible attributes of a service may shape a user's perception of quality. The attributes are independent so that inability to meet user expectations with respect to any one of them cannot be offset by exceeding user expectations with respect to others. In practical terms, this means that effective measurement of QoS will necessarily involve a collection of measures, rather than "the" measure of QoS, to serve as a basis for gauging likely user perception of service quality.

The other complication of the notion of "quality" is one of perspective. The essential distinctions can be seen in Figure 4.1. When looking at these factors there are at least three distinct, but interrelated notions of "quality of service" that might come into play in the evaluation: intrinsic, perceived and assessed QoS.

4.1.1 Intrinsic QoS

Intrinsic QoS belongs to service features from technical aspects. Thus, intrinsic quality can be determined by a transport network design and provisioning of network access, terminations and connections. The required quality can achieved using the right transport protocols and QoS assurance mechanisms. It can be easily evaluated by the comparison of measured and expected performance characteristics. User perception does not influence the intrinsic QoS rating.

4.1.2 Perceived QoS

Perceived QoS reflects the customer's experience using a specific service. It is influenced by the customer's expectations compared to the observed performance of the service. The user's expectations are affected by his experience of similar services he has used and the opinions of other customers. Thus, the QoS with the same intrinsic features may be perceived differently by various customers. So only ensuring the technical parameters of a service is not enough. Also

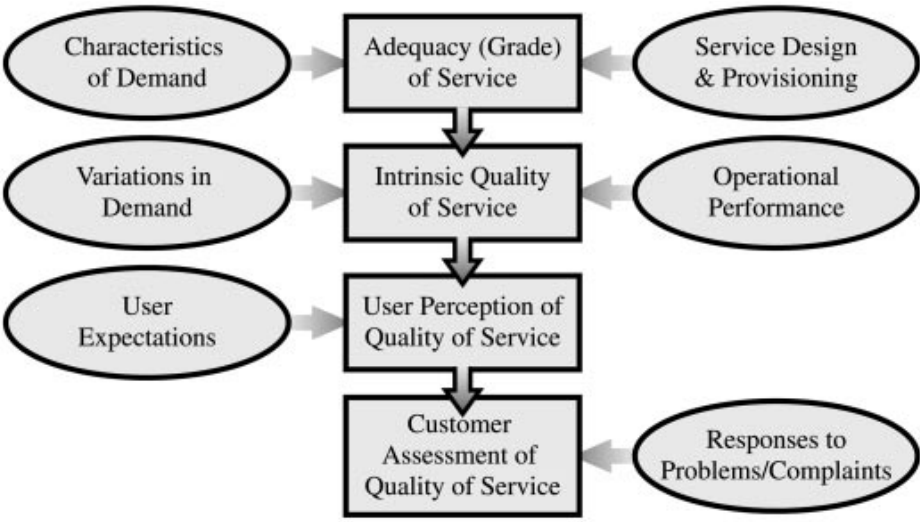

Figure 4.1: Simplified Model of Factors that Shape Perception of Quality of Service [Har01]

non-technical parameters which are important to the user have to be considered and are finally relevant for particular expectations.

4.1.3 Assessed QoS

Assessed QoS starts to be seen when a customer decides whether to continue using the service or not. This depends on the perceived quality, service price and responses of the provider to submitted user complaints and problems. It follows that even a customer service representative's attitude to a client may be an important factor in rating the assessed QoS.

4.1.4 Assurance of Satisfactory Level

The assurance of satisfactory level of the intrinsic, perceived and assessed QoS is considered separately. Assuring the intrinsic QoS lies in the responsibility of the network provider and depends on network architecture, planning and management. It is mainly a technical problem dealt with by engineers, designers and operators. To assure the perceived QoS the intrinsic QoS capabilities have to be adjusted to each particular service offered and the users have to be questioned about their experiences using the service. The assessed QoS mainly depends on the charging policy of a provider, reliable customer service representatives and technical support.

4.2 QoS in Different Domains

4.2.1 IP Networks

As noted in [GJS03] there are for IP networks three concepts to QoS definition: The International Telecommunications Union (ITU), European Telecommunications Standards Institute (ETSI) and the Internet Engineering Task Force (IETF) approach.

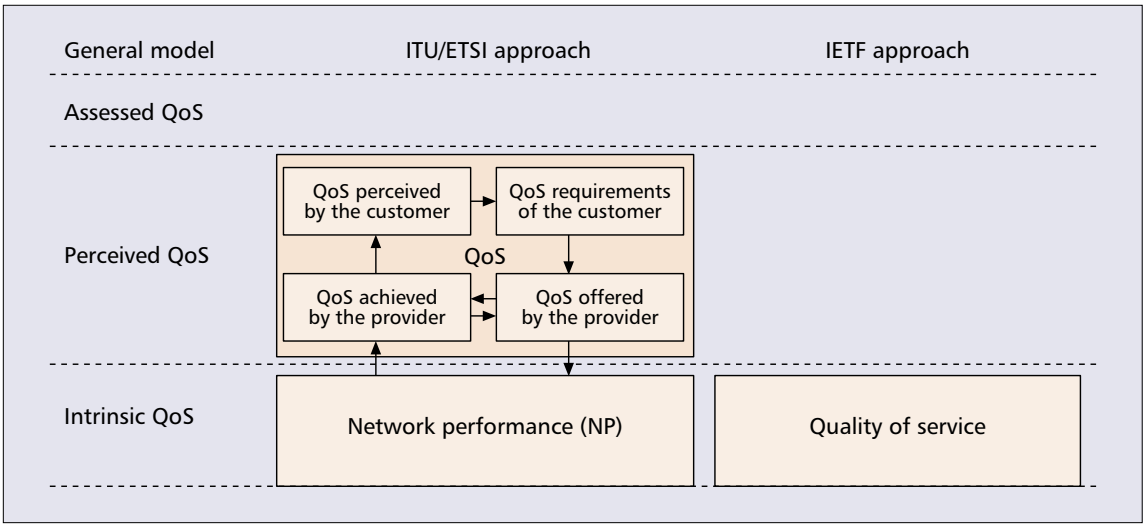

Figure 4.2: Comparison of QoS Approaches for IP Networks [GJS03]

Based on the generic QoS model which was presented in Chapter 4.1 these three approaches can be categorized as in Figure 4.2.

ITU/ETSI Approach

The ITU and ETSI approaches are almost the same. Both organisations adopted the concept of each other while developing their standards. They use the same basic definition of QoS as:

> "The collective effect for service performance while determine the degree of satisfaction of a user of the service." [ITU]

That definitions suggests that QoS in the ITU/ETSI approach concentrates mainly to perceived rather than intrinsic QoS. They introduce the notion of network performance (NP) to cover technical facts. There is a clear distinction between QoS, understood as something focused on user-perceivable effects, and network performance, which contains all necessary network functions to provide the service. From this follows that QoS parameters are user-oriented and do not directly translate into network parameters. On the other side network parameters can determine the quality which is perceived by the customers but they do not have to be meaningful for them.

Network performance corresponds to intrinsic QoS. It is defined as:

> "The ability of a network or network portion to provide the functions related to communications between users." [ITU]

NP is defined and measured in terms of parameters of particular network components involved in providing a service. A high level of NP can be achieved by appropriate system design, configuration, operation and maintenance. To cover specific points of views of QoS, the both approaches distinguish four particular definitions:

- QoS requirements of the customer
- QoS offered by the provider

- QoS achieved by the provider
- QoS perceived by the customer

The requirements of the customers state their preferences for a particular service quality. These constraints can be expressed in technical or nontechnical language which is understandable to both - the customer and the service provider. But the QoS offered by the service provider may not always meet the requirements of the customer and may be influenced by the provider's strategy, benchmarking, service deployment cost, and other factors. This is expressed by parameters understandable to the customer, e.g. "Service availability of 99.9% per year". The QoS achieved can then stated by the same set of parameters. The comparison of both - specified and achieved QoS - will result in a preliminary rating to the perceived QoS. But the most important rating are the QoS perceived by the customer, who finally rates the service quality according to his own experiences.

QoS and NP are interrelated. Ensuring high network performance is crucial to a successful service provision. Parameters of QoS can be categorized as network- and non-network-related. Network-related parameters can then be further translated into NP parameters. Target values of these parameters can be assigned which then will be measured and compared to the resulting values. The combination of the NP achieved and non-network-related QoS constitutes the QoS achieved.

IETF Approach

The IETF approach concentrates on intrinsic QoS and does not deal with perceived QoS. It stems from the main objectives of IETF, concerned with the Internet architecture and its development, dependability and effectiveness. QoS is understood by IETF as:

> "A set of service requirements to be met by the network while transporting a flow." [Cra98]

It is closely equivalent to the definition of network performance (NP) in the ITU/ETSI approaches.

The IETF has developed various QoS mechanisms for the QoS assurance in IP networks and the Internet. It proposed two significant network architectures as IntServ [BCS94] and DiffServ [ea98]. It standardized the Resource Reservation Protocol (RSVP) signaling protocol and also developed the notion of IP-QoS architecture as comprehensive approach to QoS and proposed several solutions.

IETF defines some architecture-independent QoS parameters as well as specific parameters of network components, such as traffic meters, packet markers, droppers or schedulers constituting a particular network architecture. There is a close relationship between these parameters and the "quality" experienced by packets.

QoS Parameters

Out of the comparison of the ITU/ETSI approaches the following QoS parameters can be found [GJS03].

Intrinsic QoS is defined by specific parameters for IP-based services:

- Bit rate of transferring user data available or target throughput that may be achieved.

- Delay experienced by packets while passing through the network. It can be considered as end-to-end relation or with regard to a particular network element.
- Jitter - variations in the IP packet transfer delay. Can be applied as end-to-end relation or for a single network element.
- Packet loss rate, defined as the ratio of the number of undelivered packets to sent ones.

These parameters describe the treatment of IP packets. They can be translated into particular network parameters which then can be used for ensuring a particular QoS level. Additionally intrinsic QoS may have the following attributes depending on the network architecture and the application demands:

- End-to-end (as the IntServ model) or limited to a particular domain for domains (e.g. the DiffServ model).
- Applied to all traffic or just a particular session or sessions
- Unidirectional or bidirectional
- Guaranted or statistical

QoS is usually an end-to-end characteristic of communication between end hosts. It should be assured along the whole path between peers, but the path may cross several other systems belonging to various network providers. Then performance of all autonomous systems contribute to the final service quality.

Parameters of perceived QoS are more difficult to define. They do not only depend on the network architecture, technique or mechanisms used to ensure service quality. They are usually expressed in different terms but should always be translatable into specific network parameters regardless of network architecture. An example of an extensive set of parameters of the perceived QoS is provided by ITU [ITU]. Parameters are grouped into four subsets regarding aspects of:

- Service support
- Service operability
- Service servability
- Service security

Service support reflects the provider's ability to provide a service and assist in its utilization. Parameters related to service operability determine the service ability to be operated by a user. Servability related parameters determine the service ability to be obtained when requested by the user and continue to be provided without excessive impairment for a requested duration. Service security specifies the level of a service's protection against unauthorized monitoring, fraudulent use, natural disaster and other impairments [ITU].

4.2.2 Web Services

Web services caused a paradigmatic shift from a Web of manual interactions to a Web of programmatic interactions. The work in [ZBN+04] and [LNZ04] concentrates on the choice between different web services according to their functionality and different QoS. To enable this quality-driven web service selection a framework is presented. They also go into the issue of computation and optimization by Local Optimization and Global Planning approaches which are out of the scope

of this thesis.

The presented criteria can also be used for Composite Services. The difference is that the aggregation functions for the following QoS criteria consider the whole execution path rather than only one execution step.

The proposed QoS model is extensible and includes generic and domain- or business-specific criteria. The generic criteria are applicable to all web service, for example, their pricing and execution duration. New criteria can be added to the existing model without fundamentally altering the underlaying computation mechanism. It is possible to extend the quality model to integrate non-functional service characteristics or other service QoS metrics.

Generic Criteria

As generic criteria which can be measured objectively for elementary services three parameters are considered: execution price, execution duration, and reputation. Criteria such as availability and reliability is not required in the model in [ZBN+04] due to the use of active user feedback and execution monitoring. The following parameters are proposed:

Execution price This is the amount of money which a service requester has to pay to the service provider to use his web service. Web service providers either directly advertise the execution price of their services, or they provide means for potential requesters to inquire about it.

Execution duration The execution duration measures the expected delay in seconds between the moment when a request is sent and the moment when the service is rendered. It consists of the progressing time and the transmission time.

Reputation The reputation is a measure for the service's trustworthiness. It mainly depends on the user's experiences of using a particular service. Different end users may have different opinions on the same service. The value of the reputation is defined as average ranking given to the service by end users.

Additional Criteria

Additionally to these criteria the work in [ZBN+04] contains two additional criteria for elementary services:

Successful execution rate The successful execution rate is the probability that a request is correctly responded. That means the operation is completed and a notification message of the completion has reached the end-user correctly within the maximum expected time-frame indicated. This measure is related to hardware and/or software configuration and the network connection between service requester and service provider. The value of the success rate can be computed from data of past invocations.

Availability The availability of a service is the probability that the service is accessible. It is computed as the total amount of time a service was available over the total duration of the monitoring time span. This time span can vary depending on the particular application. For example in applications where services are more frequently accessed, a small value give a more accurate

approximation for the availability of the service. If the service is less frequently accessed using a larger time-frame is more appropriate.

Business-Related Criteria

But there is also business-related criteria involved which can vary in different domains. For example in the application chosen in [LNZ04] the following criteria is used:

Transaction support This is used for maintaining data consistency. In prior QoS models, no transactional criteria is being used in the computation of the QoS value. But for some uses it could be important if a Web service provides an undo procedure to rollback the service execution or not. This transactional property can be evaluated by two dimensions: whether an undo procedure is supported and what the time constraints on undo procedures are.

Compensation rate The compensation rate indicates the percentage of the original execution price that will be refunded when the service provider can not meet the committed service quality.

Penalty rate It indicates what percentage of the original price service requesters need to pay to the provider when they want to cancel the committed service or ordered commodity after the time out period for transaction to roll back is expired.

4.2.3 Workflow Systems

[CSM+04] presents a comprehensive model for the specification of workflow QoS as well as methods to compute and predict QoS. It does not only target the time dimension, but also investigate other dimensions required to develop an usable workflow QoS model. In a second part also the needed enhancements are described which are needed for existing workflow systems to support processes constrained by QoS requirements. The enhancements include the implementation of a QoS model, the implementation of algorithms to compute and predict workflow QoS, and the implementation of methods to record and manage QoS metrics.

Workflow QoS represent the quantitative and qualitative characteristics of a workflow application necessary to achieve a set of initial requirements. Quantitative characteristics can be evaluated in terms of concrete measures such as workflow execution time, cost, etc. Qualitative characteristics specify the expected services offered by the system, such as security and fault-tolerance mechanisms.

Workflow Characteristics

One of the most popular workflow classifications distinguishes between ad hoc workflows, administrative workflows, and production workflows. This classification was first mentioned in [McC92]. The main differences between these types include structure, repetitiveness, predictability, complexity, and degree of automation.

The presented QoS model in [CSM+04] is better suited for production workflows since they are more structured, predictable and repetitive. Production workflows involve complex and highly-structured processes, whose execution requires a high number of transaction accessing different information systems. In the case of ad hoc workflows the information, the behavior, and

the timing of tasks are largely unstructured, which makes constructing a good QoS model more difficult.

QoS Dimensions

Task time Time is a common and universal measure for performance. Time is critical competitive advantage for most businesses: Shorter workflow execution time allows a faster production of new products. The first measure of time is task *response time (T)*. Task response time corresponds to the time an instance takes to be processed by a task. It can be broken down into two components: *delay time (DT)* and *process time (PT)*.

$$T(t) = DT(t) + PT(t)$$

Delay time (DT) refers to the non-value-added time needed for an instance to be processed by a task. This includes the instance queuing delay and setup time of the task. Those two metrics are part of the task operation, but they do not add any value to it. It can be further broken down into *queuing delay* and *setup delay*. *Queuing delay* is the time instances spend waiting in a task list, before they are selected for processing. *Setup delay* is the time an instance spends waiting for the task to be set up. Another time metric that may be considered is the *synchronisation delay*, which corresponds to the time a workflow instance waits for other instances in an and-join task (synchronisation). This metric is not part of this proposed QoS model.

Process time (PT) is the time a workflow instance takes at a task while being processed. In other words, it corresponds to the time a task needs to process an instance.

Task cost Task cost represents the cost associated with the execution of workflow tasks. Prior to workflow instantiation and during workflow execution, it is necessary to estimate the cost of the execution in order to guarantee that financial plans are followed. The cost of executing a task includes the cost of using equipment, the cost of human involvement, and any supplies and commodities needed to complete task. *Task cost (C)* is the cost charged when a task t is executed. It can be broken down into two components: *enactment cost* and *realization cost*.

$$C(t) = EC(t) + RC(t)$$

Enactment cost (EC) is the cost associated with the management of the workflow system and with the monitoring of workflow instances.

Realization cost (RC) is the cost associated with the runtime execution of the task. It can be broken down into: *direct labor cost*, *machine cost*, *direct material cost*, *and setup cost*. *Direct labor cost* is the cost associated with the person carrying out the execution of a workflow human task, or the cost associated with the execution of an automatic task with partial human involvement. *Machine cost* is the cost associated with the execution of an automatic task. This corresponds to the cost of running a particular piece of software or the cost of operating a machine. *Direct material cost* is the cost of the materials, resources, and inventory used during the execution of a workflow task. *Setup cost* is the cost to set up any resource used prior to the execution of a workflow task.

Task reliability The model for the reliability dimension of workflows in [CSM+04] is based on concepts from system and software reliability theory. *Task reliability (R)* models what can be considered the most important class of workflow failures and task failures (also known as activity

failures). Task failures can be organized into two main classes: *system failures* and *process failures*.

System failures consist of information technology and software failures which lead to a task terminating abnormally.

Process failures consist of business process exceptions which lead to an anomalous termination of a task. As presented in [KS95] in a workflow, task structure has an initial state, an execution state, and two distinct terminating states. For non-transactional tasks, one of the terminating states indicates that a task has failed, while the other state indicates that a task is done (see Figure 4.3). For transactional tasks, the terminating states are aborted and committed. That means, that only one starting point exists when performing a task, but two different states can be reached upon its execution.

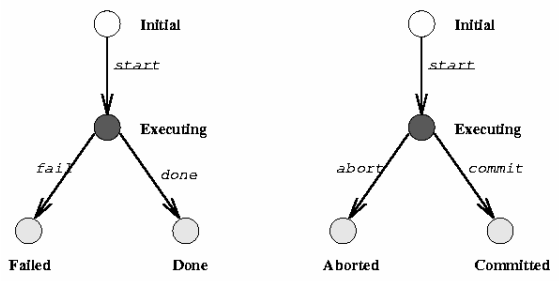

Figure 4.3: Workflow Task Structures [KS95]

Task reliability is described using a discrete-time modeling approach. This approach was selected, since workflow task behavior is mostly characterized in respect to the number of executions. It can be computed as follows:

$$R(t) = 1 - (systemfailurerate + processfailurerate)$$

System failure rate is the ratio between the numbers of time a task did not perform for its users and the number of times the task was called for execution. *Process failure rate* provides information concerning the relationship between the number of times the state done/committed is reached and the number of times the failed/aborted state is reached after the execution of a task.

Alternatively, continuous-time reliability models can be used when the failures of the malfunctioning equipment or software can be expressed in terms of times between failures, or in terms of the number of failures that occurred in a given time interval.

4.2.4 Peer-to-Peer Networks

Peer-to-Peer computing grids consist of peer nodes that communicate directly among themselves through wide-area networks and can act as both clients and servers. [GN02] describes a scalable QoS service aggregation model for Peer-to-Peer computing grids. Peer-to-Peer networks are attractive since they promote resource sharing such as sharing of processing cycles and disk storage, without any administration cost or centralized infrastructure support.

Most of the proposed approaches for QoS models for Peer-to-Peer networks present the following major limitations. First, they lack generic QoS support for coordinating arbitrary interactions between service instances. Second, they do not provide dynamic peer selection scheme

since all services are assumed to be provided by dedicated servers. Third, they often assume a global view of the entire system in terms of performance information, which is impossible in Peer-to-Peer system due to the scalability requirement. Fourth, they do not consider the dynamic topological variation caused by arbitrary peer arrivals/departures in Peer-to-Peer systems.

Due to the inherent redundancy property of Peer-to-Peer systems, a service instance can also have multiple replications, which are provided by different peers. The system has to decide the specific peers where the service instances are actually executed. To consider topological variation, caused by arbitrary peer arrivals/departures, peer's uptime is proposed to be used as one of the selection metrics.

A QoS-aware service aggregation (QSA) model should meet the following challenges:

1. Decentralization: The solution must be fully distributed and only involve local computation based on local information.
2. Scalability: The solution must scale well in the presence of a large number of peer nodes.
3. Efficiency: The solution should be able to utilize resource pools provided by the Peer-to-Peer systems efficiently so that it can admit as many user requests as possible.
4. Load balance: Although each peer makes decisions based on only local information, the solution should achieve the desired global properties such as load balance in Peer-to-Peer systems.

4.3 Summary

After the discussion of several QoS models and approaches, we will now compare the different approaches and specify requirements for a QoS model. Based on these specified service attributes we will then construct our QoS model in Chapter 5.

4.3.1 Comparison

If we compare the different QoS approaches and domains where QoS is used today, we can outline the following:

Generic QoS Model

The generic QoS model in Section 4.1 provides an useful scheme for categorizing QoS criteria. It is not only focused on the technical aspects (intrinsic QoS) of a service, but also on the requester's view (perceived and assessed QoS). This model allows us to describe every-day services in a matter which is close to the end-user's perspective.

IP Networks

The QoS approaches in the domain of IP networks are mainly focused on technical aspects (see Section 4.2.1). Both, the ITU as well as the ETSI model, introduce the term of *network performance*, which covers the technical aspects of a network (such as system design, configuration, operation and maintenance). To cover specific points of views of QoS, both models distinguish between: QoS requirements of the customer, QoS offered by the provider, QoS achieved by the provider

and QoS perceived by the customer. The different parameters which can be gained of the these two approaches are listed in Section 4.2.1.

The IETF approach on the other side concentrates on intrinsic QoS and does not include perceived QoS. Its definition of QoS is closely equivalent to the definition of network performance in the ITU/ETSI approach.

Web Services

As outlined in Section 4.2.2 there are several papers which concentrate on the choice between different Web services by their functionality and QoS. As noted there the topics of Local Optimization and Global Planning are out of the scope of this thesis. The presented QoS criteria can also be used for Composite Services. The proposed QoS model contains generic and domain- or business-specific criteria. It is also extensible according to the actual needs of a service.

The presented open, extensible framework is a handy approach which we will use. Its QoS parameters can be almost completely mapped to our workflow system. But with one exception, availability is not appropriate, since in a Peer-to-Peer environment users can disappear and reappear at any time.

Workflow Systems

The QoS dimensions in Section 4.2.3 are specially adapted to Workflow systems. The different parameters such as task time, delay time and task cost, and their further fragmentations can be integrated in our model.

Peer-to-Peer Networks

Finally, the comments in Section 4.2.4 show the specialties of QoS in Peer-to-Peer environments. Peer-to-Peer networks are a bit different: First, they cannot provide a global view of the entire system because of scalability issues. Second, peers can arrive and depart at any time. This must be considered when a service is requested. Additionally the approach may only involve local computation based on local information. Further, each peer's uptime can be used as a selection criteria between different peers.

4.3.2 QoS Requirements

Out of the comparison of different QoS approaches in Section 4.3.1, we can identify the key features of a QoS model for workflow systems in Peer-to-Peer environments that meets our needs. These features are:

Extensible Criteria

The criteria can consist of different non-functional properties, as presented in Section 2.2. To ensure that the end-user's perspective is taken into account, the business aspects should be in focus, not only technical aspects. The QoS model has to meet the following terms:

First, the criteria must be based on the generic QoS model which was presented in Section 4.1. This allows us to distinguish between intrinsic QoS, perceived QoS and assessed QoS. For perceived and assessed QoS criteria we can use active user feedback through ratings.

Second, to include domain- and business-specific criteria the model has to be extensible. This can be achieved using RDF technology, which will be the base technology for modeling our QoS model and ontology. Our approach will include some generic criteria (such as execution price, execution duration, and reputation) which is feasible for all kinds of services. If additional specific criteria is needed, this can be added through the extension of the given QoS base model.

Third, the chosen criteria has to be optimal. That means it can be collected easily and cost-effective, as outlined in the specification process in Section 3.2.1.

Support for Different Decision-Makers

As mentioned in Section 3.1.1, there are different groups of decision-makers which will use QoS. The actual users of the service may other concerns than the system architects and engineers. They can have a totally different perception of QoS and the evaluation has to be different for each case.

For each single service, service requester and service provider may have different views of the QoS. To compare several service providers, each criteria could be weighted on a scale from 0 to 5. Where 0 means not needed, and 5 gives the highest ranking. A comparison of different service providers could then be easily achieved by adding the products of each criteria's rating and value. This method is only applicable for numeric parameters, therefore we will not use it for our model.

Support for Workflow Systems

The QoS model has to involve QoS criteria which meets the requirements of Workflow systems. Especially the time, cost and reliability parameters must have the sufficient granularity to measure the performance of the workflow.

The formulation of workflows using the BPEL description language brings along that temporal representation as stated in Section 2.2.1 can be expressed (e.g. service X begins after service Y). Additionally the settlement process as described in Section 2.2.4 is also supported: The mutual obligations of the provider (service delivery) and requester (usually payment) can be modeled using BPEL's input and output document approach.

Support for Peer-to-Peer Systems

Since our QoS will be used in a Peer-to-Peer environment, it has to support the corresponding requirements. Active monitoring of the peers involved will not be used to prevent an unnecessary network overload. One could use the peer's uptime additionally to the other selection metrics.

Security and Trust

As soon as real-world services will be provided through our service platform, security becomes an issue. In our QoS model we will not implement any kind of security mechanism. Trust on the other side is crucial, it can be easily implemented using reputation mechanisms, such as the one described for Web services (Section 4.2.2) or Workflow systems (Section 4.2.3).

Chapter 5

QoS Model and Scenarios

In the following sections the proposed QoS model is introduced step by step and adapted to our requirements. Each model is described using RDF Schema [BG04] in combination with basic data types provided by XML Schema [BM01]. The complete RDF representation listings can be found in the appendix in Section A.

5.1 Basic QoS Model

Based on the requirements obtained from comparing different QoS approaches in Section 4.3 we will now compose our QoS model. It will be modeled using RDF technology [BM04] which also allows to include some basic data types of XML Schema [BM01].

Criteria

Our basic model includes parameters to model cost, delegation, reputation, and time of a workflow task. Table 5.1 shows a graphical representation.

Description	Data type
Cost	xsd:float
Delegation	xsd:boolean
Reputation	-
Process failures	xsd:positiveInteger
Rating	xsd:positiveInteger
System failures	xsd:positiveInteger
Time	-
Delay time	xsd:positiveInteger
Task time	xsd:positiveInteger

Table 5.1: Basic QoS Model

The following parameters are involved, each one's data type is noted in brackets.

Cost How much a specific service costs (data type: float).

Delegation Shows whether delegation of a task is allowed or not (data type: boolean).

Reputation Reputation is a combined criteria consisting of:

- *Process failures:* Number of process failures that occurred during the execution of a process or task (data type: positive integer).
- *Rating:* Indicates how satisfied a service requester was with the service. Rated on a scale from 1 to 5, where 1 means totally unsatisfied and thus 5 totally satisfied. (data type: positive integer).
- *System failures:* Number of system failures that occurred during the execution of a process or task (data type: positive integer).

Time To support in particular workflow process (see Section 4.2.3), time is split up into:

- *Delay time:* Delay time includes queuing and setup delay of a specific task (data type: positive integer, time in milliseconds).
- *Process time:* Process time is the time a task takes to be executed (data type: positive integer, time in milliseconds).

5.2 Scenarios

We will use the following real-world scenarios to show the extensibility of the proposed QoS model. For that we use a shorter version of the specification process described in Section 3.2. First, the audience which uses the QoS criteria and the objectives of the actors in our workflow process will be described. Second, we will specify additional QoS criteria.

5.2.1 Scenario A - Pizza Service

The first scenario deals with a common pizza service, where people can order pizzas from and get them delivered to their homes.

Audience and Objectives

For a pizza service the audience consists of two parties:

Customer The customer wants the best pizza he can find, as fast as possible. He is interested in the total waiting time (*delay time* + *process time* + *delivery time*) and the range of different *pizza toppings*. Furthermore the *pizza rating* by other customers is an useful indicator what quality one can expect. Finally, also *billing* issues are important.

Pizza shop The personnel and the owner of the pizza shop are interested if a pizza meets the customers's expectations. Additionaly to the rating in the reputation parameter, which describes the overall rating of the service, *pizza rating* is introduced. To monitor and improve the service performance the fine granularity of the time parameter is fundamental: It can even be extended by *delivery time* as third parameter.

Description	Data type
Billing	none, can contain further RDF resources
Cost	xsd:float
Delegation	xsd:boolean
Pizza rating	xsd:positiveInteger
Pizza toppings	none, can contain further RDF resources
Reputation	-
Process failures	xsd:positiveInteger
Rating	xsd:positiveInteger
System failures	xsd:positiveInteger
Time	-
Delay time	xsd:positiveInteger
Delivery time	xsd:positiveInteger
Task time	xsd:positiveInteger

Table 5.2: QoS Model of Scenario A - Pizza Service

Criteria

This scenario involves the following criteria as shown in Table 5.2 (scenario-specific parameters are highlighted in *italic*).

The extensibility of RDF permits to define open criteria properties. For example, we could define a shop-specific RDF ontology of available pizza toppings. Using the basic RDF constructs as `rdf:Bag` these resources can directly be refered to.

The extend QoS model of our pizza scenario (see Table 5.2) contains additionally to the basic criteria in Section 5.1 the following parameters:

Billing Contains a collection of billing alternatives such as cash or credit card, given as strings (data type: none, can contain further RDF resources).

Pizza rating Rating of the pizza quality on a scale from 1 to 5, where 1 means totally unsatisfied and thus 5 totally satisfied. (data type: positive integer).

Pizza toppings Contains a collection of different toppings a customer can select from, specified as strings (data type: none, can contain further RDF resources).

Time The time parameter is extended by:

- **Delivery time:** Time required to deliver a finished pizza from the pizza shop to the customer's home (data type: positive integer, time in milliseconds). To allow a comparison of different delivery times, this parameter needs to be divided by the distance in km.

5.2.2 Scenario B - Online Music and Video Store

This scenario contains an online music and video store where Internet users can download from songs or movies in different formats and qualities.

Audience and Objectives

We can distinguish between the following actors:

Internet user To make a selection of different online music and video stores, Internet users are mostly interested in the quality of the provided goods. For movies the *movie dimensions* (*movie width* and *height*) and the *movie frame rate* are important. For songs on the other side, the *song bit rate* is important. Additionally to that the products property delivers information about how big the range of products is (*number of movies* or *number of songs*) or how popular a service is (*number of downloads*).

Service provider The service provider is mainly interested how popular his service is (*number of downloads*). It would be interesting to have this number itemized by each single movie or song. To simplify matters we will just consider the total number of downloads.

Criteria

This scenario contains the following service parameters (scenario-specific parameters are highlighted in *italic*):

Description	Data type
Cost	xsd:float
Delegation	xsd:boolean
Movie quality	-
Movie dimensions	-
Movie height	xsd:positiveInteger
Movie width	xsd:positiveInteger
Movie frame rate	xsd:positiveInteger
Products	-
Number of downloads	xsd:positiveInteger
Number of movies	xsd:positiveInteger
Number of songs	xsd:positiveInteger
Reputation	-
Process failures	xsd:positiveInteger
Rating	xsd:positiveInteger
System failures	xsd:positiveInteger
Song quality	-
Song bit rate	xsd:positiveInteger
Time	-
Delay time	xsd:positiveInteger
Task time	xsd:positiveInteger

Table 5.3: QoS Model of Scenario B - Online Music and Video Store

Since this service will be provided in a Peer-to-Peer environment, the traditional QoS parameters from the domain of IP networks (such as streaming bit rate and jitter) can not be directly applied to this service. This example also shows how easily composite QoS parameters such as *movie quality* can be expressed using RDF.

The QoS model for this scenario involves the following parameters:

Movie quality Describes the quality of movies by meaning of

- **Movie dimensions:** Width and height of a movie in pixels (data type: positive integer).
- **Movie frame rate:** Movie frame rate in frames per second (data type: positive integer).

Song quality Describes the quality of songs by meaning of

- **Song bit rate:** Bit rate of the song in kilobytes (data type: positive integer).

Products To describe the range of products and their popularity we use

- **Number of downloads:** Indicates how many times a movie or a song was downloaded (data type: positive integer).
- **Number of movies:** Total number of movies to choose from (data type: positive integer).
- **Number of songs:** Total number of songs to choose from (data type: positive integer).

5.2.3 Scenario C - Supply-Chain Management

This scenario represents a classic supply-chain management process in the engineering industry. An assembler orders required elements from a distributor who deals with two manufacturers.

Audience and Objectives

As stated above in this scenario we have three types of actors:

Assembler The assembler is mainly interested in the quality of the delivered goods (*production failure rate*) and the overall delivery time (*delivery time* + *production time*).

Distributor The distributor needs to have information about the *production time* of a specific good. Furthermore, he has to know if this task can be delegated to another company (*delegation*).

Manufacturer Depending on the *priorization* of an order the manufacturer will immediately produce the ordered item or queue it. For optimizing the production process measures as *production failure rate* and *production time* might be useful.

Criteria

A scenario of a supply-chain environment could consist of the criteria shown in Table 5.4 (scenario-specific parameters are highlighted in *italic*).

Additionally, some business-specific parameters are added, such as *penalty rate* which describes the amount of money the provider is obligated to pay in case of delay. In this case delegation is allowed, so the delegation flag will be set to true. The criteria involves:

Penalty rate The amount of money the service provider has to pay if the agreed delivery time cannot be met (data type: positive integer).

Description	Data type
Cost	xsd:float
Delegation	xsd:boolean
Penalty rate	xsd:positiveInteger
Reputation	-
Process failures	xsd:positiveInteger
Rating	xsd:positiveInteger
System failures	xsd:positiveInteger
Time	-
Delay time	xsd:positiveInteger
Delivery time	xsd:positiveInteger
Production time	xsd:positiveInteger
Task time	xsd:positiveInteger

Table 5.4: QoS Model of Scenario C - Supply-Chain Management

Production failure rate Percentage of produced items which is defective or does not meet the specified requirements (data type: float).

Priorization This parameters allows to indicate the priority of a specific order. It can e.g. be used to differ if the manufacturer deals with a common customer or a company which has special conditions.

Time The time parameter is extend by

- **Delivery time:** Time required to deliver an ordered item from the storehouse to the clients' company (data type: positive integer, time in milliseconds). To allow a comparison of different delivery times, this parameter needs to be divided by the distance in km.
- **Production time:** Time required to produce an item (data type: positive integer, time in milliseconds).

Chapter 6

Implementation

The following chapter deals with the implementation of the proposed QoS model for Peer-to-Peer workflows into the MOTION [KFRG02] platform. First, the architecture of the MOTION middleware is shortly explained, and after that the MOTION workflow component which was contributed by Daniel Schwarz [Sch04]. Finally, the implementation of the QoS component is described.

6.1 MOTION

The MOTION (MObile Teamwork Infrastructure for Organisations Networking) [KFRG02] middleware is a platform for mobile teamwork and collaboration. The MOTION service architecture supports mobile teamwork by taking into account the different connectivity modes of users, provides access support for various devices such as laptop computers and mobile phones, and uses XML meta-data and the XML Query Language (XQL) for distributed searches and subscriptions.

MOTION uses the following terminology:

- *Artifact*: Any document or file in the MOTION system.
- *Peer*: Any computing device connected to the MOTION system.
- *Community*: A collection of users in the MOTION system that are interested in a specific topic and have a common property.

6.1.1 Architecture Overview

The MOTION middleware has a layered architecture as shown in Figure 6.1. The system is composed of peers which act as peers and can host services. MOTION configurations can consist of desktop computers, laptops, PDAs that host services, and clients such as Web browsers and WAP-enabled mobile phones, that are only used to remotely access the provided services.

The lowest layer of the architecture is the communication middleware. It provides basic communication services as Peer-to-Peer file sharing through distributed search and publish/subscribe mechanisms to the layers above. In the current implementation of MOTION this functionality is provided by Peerware [CP01].

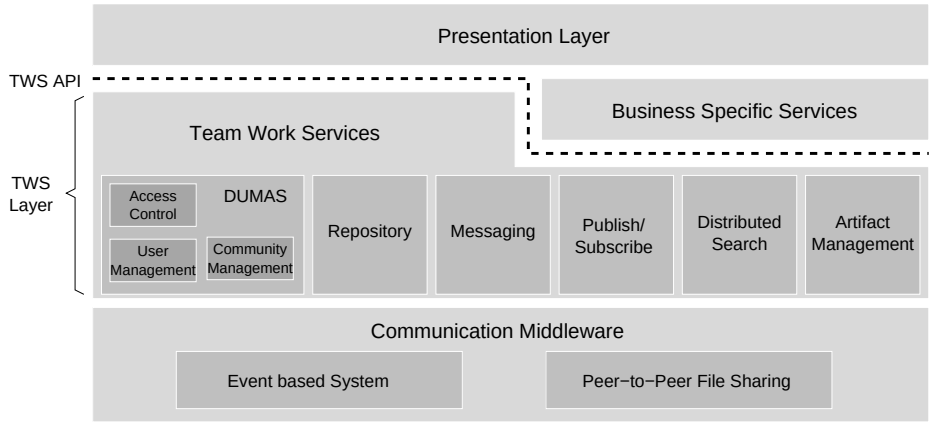

Figure 6.1: Overview of the MOTION Architecture [KFRG02]

The Teamwork Services (TWS) layer is situated directly above the communication middleware. This layer integrates the basic system components such as the repository and DUMAS (see Section 6.1.2) and provides the Application Programming Interface (API) to the teamwork services. The TWS API provides services such as:

- storing artifacts and their meta-data (profile) in the local repository,
- managing resources (artifacts, users, and communities),
- sharing artifacts with other users in communities,
- subscription to specific events in the MOTION system,
- sending and receiving messages from other users or from the system,
- managing access rights on resources,
- and searching for resources based on their profile information.

The top layer of the architecture is the presentation layer which provides an user interface to the services provided. This layer is using the TWP API.

6.1.2 Teamwork Services Components

The MOTION Teamwork services layer consists of the following components (see Figure 6.1):

Dynamic User Management and Access Control Component (DUMAS) component

DUMAS [Fen00, FGRK01] is the access control component of the system. It is a generic, customizable component that satisfies different security requirements. DUMAS provides support for managing users and roles (e.g., by creating, deleting, etc.) and assigning users to roles. For performing any security sensitive operation (e.g. creating a user, downloading an artifact, etc.) it must be consulted to find out whether the user is permitted to invoke this operation.

DUMAS can also customized to run on a variety of mobile devices.

Messaging Component

The integrated MOTION Messaging service enables users to communicate and exchange information. Notifications based on subscriptions to the offered topics are also delivered by this service. It distinguishes between the following types of messages: *System-to-User*, *System-to-Community*, *User-to-User*, and *User-to-Community* messages.

- *System-to-User* and *System-to-Community* messages are sent by the MOTION system as notifications of subscriptions.
- *User-to-User* messages are sent by users to other users.
- *User-to-Community* messages are sent by one user to a community. In the MOTION system each user is referenced by his login name. The users can also specify the medium under which they are available.

The MOTION messaging component consists of five main components: the SMS gateway, the SMTP gateway, the standard messages gateway, the WAP gateway, and the MOTION frontend component (see Figure 6.1). The MOTION frontend component builds the interface between business processes and the MOTION messaging component.

Publish/Subscribe Component

This service bridges the gap between the underlaying middleware and the business-specific services. Publish/subscribe systems such as Peerware [CP01] allow components to subscribe and react to specific events. In Peerware the subscriber specifies an object for callback.

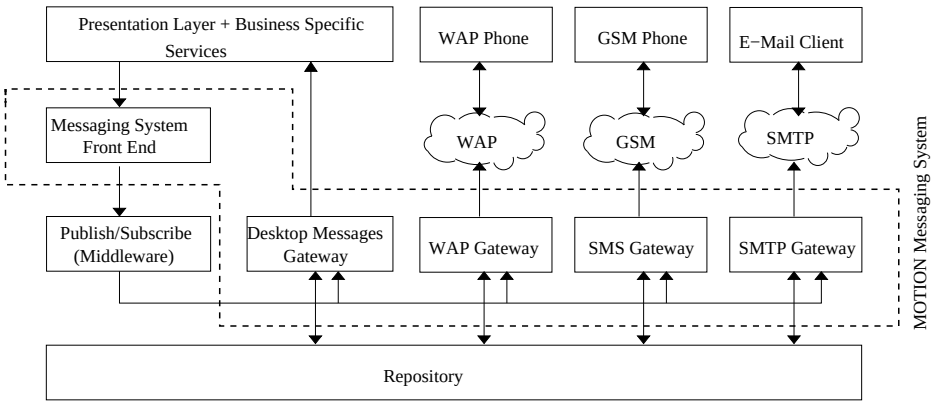

Figure 6.2: Teamwork Services Layer Publish/Subscribe Component Architecture [KFRG02]

In MOTION this functionality is implemented through *subscription gateways* and *user specialized callbacks* (see Figure 6.2). *A user specialized callback* is a component that handles subscriptions of a specific user. It handles the user's subscriptions and informs the single callbacks if the specified event occurs. The set of callback components running on a particular peer is referred to as *subscription gateway*.

To hide differences between different middleware platforms the XML Query Language (XQL) is used to bridge between business services and the middleware. This enables business processes

to query complex XML events. The Teamwork Services Layer Publish/Subscribe Component provides the business specific services with the capability of subscribing to users, artifacts, and communities in the system.

Repository Component

Every peer that runs MOTION services contains a repository that is used to store artifacts and profile information about users, communities and artifacts. It consists of two parts a *XML* and an *artifact repository*. The *XML repository* is used to store XML profile information. The *artifact repository* stores artifacts that belong to an user. In the MOTION prototype artifacts and the artifacts repository are stored on the local file system.

Inserting, editing, and deleting artifacts will be done in the user's own resource space. In order to share an artifact in a community, this artifact is tagged as belonging to the specific community and is visible to other users as long as the owner is online. If the user wishes to make an artifact persistent, it can be copied to the so called community cabinet.

Artifact Management Component

The artifact manager component consists of the MOTION *repository component* and the *repository manager*. The repository manager maps remote transfer requests to commands in the repository. It manages the information in the repository and provides the communication between peers.

The Artifact Manager component acts as a wrapper to the MOTION repository and the *repository manager*. It provides artifact management API calls in the TWS API.

Distributed Search Component

The TWS API provides mechanisms to search the artifacts that are in the MOTION system, and provides an advanced search support with a fine granularity (using XQL queries).

6.2 Workflow Component

The MOTION workflow component was implemented by Daniel Schwarz [Sch04]. It is integrated in the MOTION system (see Section 6.1).

6.2.1 Terminology

The MOTION workflow component [Sch04] uses the following terminology for modeling workflow processes:

Communities

The concept of the MOTION system is based on communities. A community is a collection of several peers (so called community members) which semantically share the same interests. A peer can be a member of several different communities, that means communities can overlap.

As noted in [Sch04], the following three notions can be distinguished (see Figure 6.3):

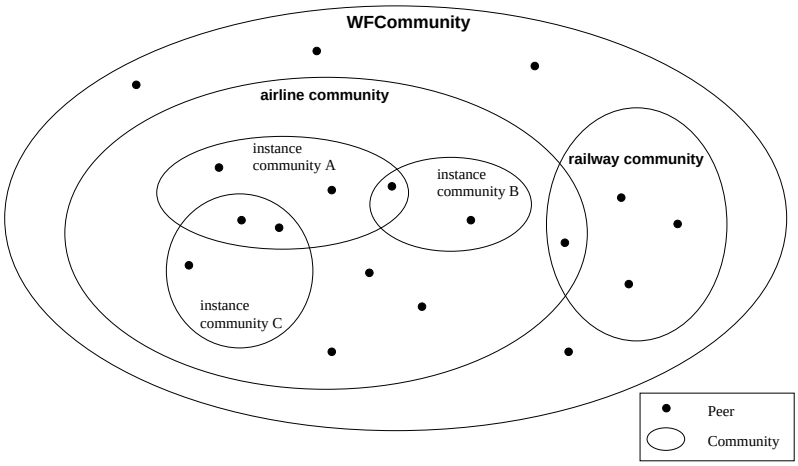

Figure 6.3: MOTION Communities [Sch04]

WFCommunity This is a meta community of the whole workflow system. All participants of single workflow communities have to be members of this top-level community. It serves as a communication platform: Messages of global interest can be announced here.

Process communities Peers with similar interests are grouped into process communities. Any peer which wants to participate in e.g. an airline reservation process would have to join the corresponding process community.

Instance communities Upon creation of a process instance an instance community will be established automatically. The peers which take part in this process execution, will become automatically members of this community.

Processes

Business processes are modeled using the BPEL description language [CGea02]. To import a new *process* into the MOTION workflow system, the peer has to choose the corresponding BPEL description file and must select a community which the process will be published in. The *Workflow Coordinator* stores that information in the *Workflow Database* and creates a MOTION artifact for the BPEL file. This artifact is added to the given community. These steps are illustrated in Figure 6.4.

Once a new process has been created and its BPEL file has been published to a community, members of this community can download the BPEL file to their local repository.

Tasks

Since processes can contain several parts, a process has to be split into single tasks. These single *tasks* can then also be distributed to different peers for completion. The BPEL process description contains detailed information about these tasks, such as:

- Dependencies between tasks: Sequence in which the tasks have to be executed.

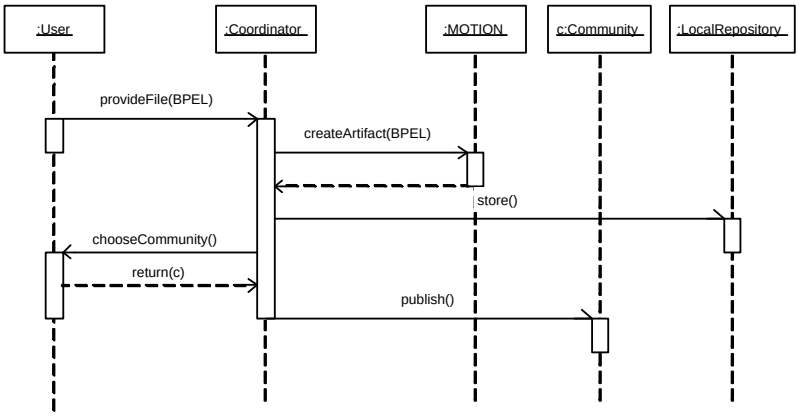

Figure 6.4: Import a New Process [Sch04]

- Required input data: Data which is either provided by the Coordinator Peer or output data of a preceding task.
- Produced output data: It can be passed to a following task or be the output data of the task.

All parts of a process are distributed to other peers for execution. If a peer wants to provide a particular task it has to announce it to the corresponding community (see Figure 6.5). This includes also the QoS parameters.

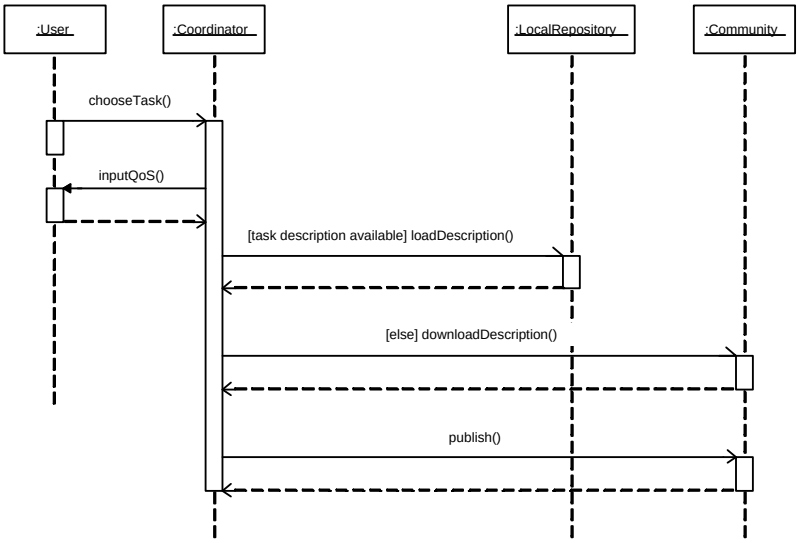

Figure 6.5: Provide a New Task [Sch04]

Instances

If a process should be executed, a peer can create an *instance* of it. The peer becomes coordinator of this instance and is responsible for the assignment of the tasks and providing the required input data (see Figure 6.6). An instance contains data concerning the execution, like

- assigned peers and details about their tasks,
- workflow status of the tasks,
- documents involved.

The peers that participate in the execution are grouped together into a private community, so called *Instance Community*.

To create an instance the coordinator has to choose a process which it was to execute. The *Workflow Coordinator* creates an instance community for this instance in which the relevant information will be exchanged. After that the new instance will be published to the process community.

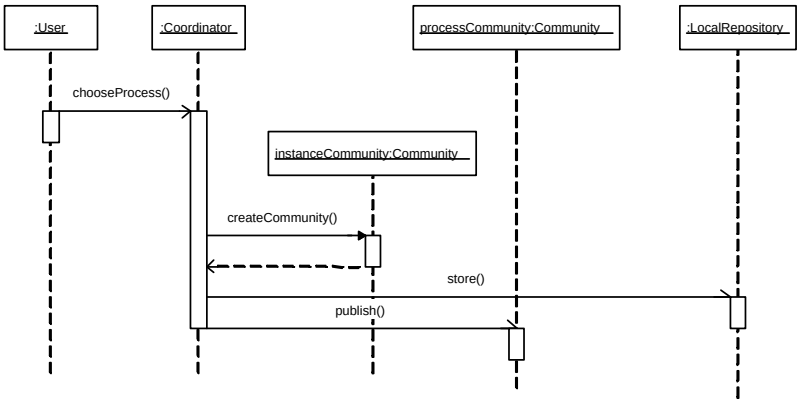

Figure 6.6: Create an Instance of a Task [Sch04]

6.2.2 Architecture Overview

In the current implementation of the MOTION workflow component lies atop the MOTION layer (see Figure 6.7).

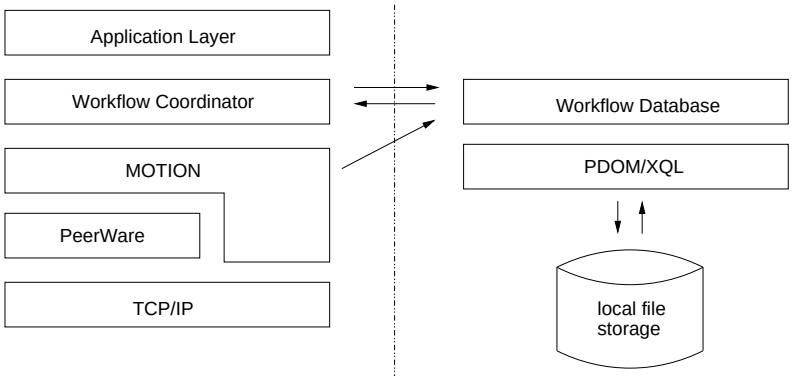

Figure 6.7: MOTION Workflow Architecture [Sch04]

The system consists of two mayor components:

Workflow Coordinator

The *Workflow Coordinator* is the central part of the workflow engine. It provides most of the functionality which is required by the application layer, such as: user management, community management, process and instance management, and communication between peers. It is tightly coupled to the *Workflow Database* where all the data is retrieved from. This ensures a loosely coupling of the peers.

Workflow Database

The *Workflow Database* stores and retrieves information of all known peers. It listens to messages of the MOTION system, and adds if necessary the relevant data to the database.

Data can be distinguished between *process data* and *user data*. *Process data* holds a description of all known processes, such as the structure of involved task and input/output data. *User data* contains a collection of all known users, including each user's announced tasks and processes instances.

The data is persistently stored in the local repository and accessed using PDOM and XQL technology.

6.2.3 Data Distribution

The communication between peers is based on message delivery and the exchange of data files wrapped into MOTION artifacts [Sch04]. The concept of communities (see Section 6.2.1) helps to ensure low bandwidth consumption and to avoid sending unnecessary messages.

Messages

Messages can be sent from one peer to another or to a community. Messages help to control the workflow process and distribute actively information to the peers. Depending on the message's scope we can distinguish the following types:

Global messages These are global announcements which are sent to the top-level community WFCommunity. There are two types of messages in this section:

- Create/Remove a process community: When a process community is created, this is announced to all members of the Workflow system. The peers can then decide whether they want to join this new community or not.
- Global search for task announcements: This is used by peers which are not permanently connected to the network. A global search queries all peers if they offer a specific task and updates the actual view of online peers.

Process Community messages These messages are sent to process communities and informs the peers which are subscribed to this community. There is one type:

- Provide/Revoke task announcement: If a peer decides to newly provide a task, to change its QoS attributes, or to stop providing a task, it has to inform the community.

Direct Peer-to-Peer messages These messages are only sent from one peer to another. They are used for communication between a coordinator of an instance and the involved peers. Two types appear:

- Negotiation of peer assignments: If a coordinator wants a peer to execute a task, it will ask the potential partner if it is willing to participate in this instance.
- Update of a task's status: A peers which is assigned to a task has to inform the coordinator of the status of this work. If a new status is reached, a notification message is sent to the coordinator peer.

Data Files

To distribute data files between peers MOTION artifacts are used. An artifact is a wrapper for data files which can be published to a specific community. The members of that community can then decide if they want to download the artifact or not. There are two cases in which data files are published to the workflow system:

Process description files Process description files are provided as BPEL files [CGea02]. A coordinator, which introduces a new process, also publishes the corresponding BPEL file to the community he wants the process to execute in. Potential process participants can then download the process description file to their local repository.

Instance data files For exchanging data involved in a workflow execution instance data files are used. Single tasks in the workflow process may need input documents and provide output documents when the task is finished. All these documents are published using MOTION artifacts into the matching instance community, where the participating peers can download them.

6.2.4 Instance Life Cycle

As it has be mentioned before, business processes are executed in process instances. There are the following steps in the lifetime of an instance.

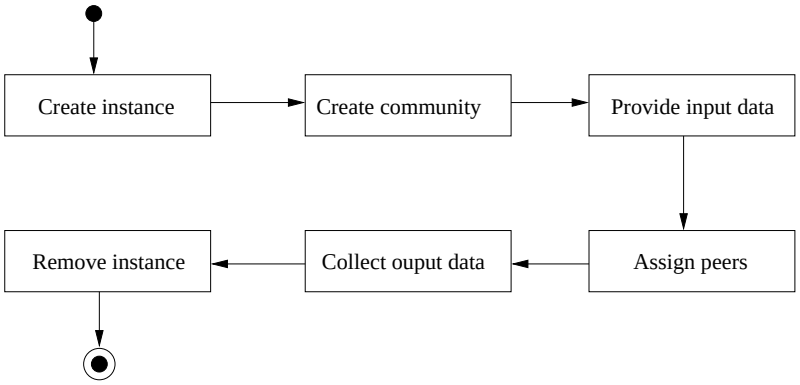

Figure 6.8: Instance Life Cycle [Sch04]

1. Create an instance: The coordinator peer invokes the instantiation by choosing the process he wants to execute.

2. Create a community: Automatically an instance community is created which contains the instances' messaging and data files.
3. Provide input data: If input data is required, it has to be supplied before the tasks can start.
4. Assign peers: For each task a peer has to be assigned which is responsible for performing the specific task.
5. Collect output data: The coordinator collects the output data which is the result of the overall work.
6. Remove an instance: After the process in completed, the instance and its instance community are automatically removed from the Workflow system.

6.2.5 Task Status

When a published task is requested by an coordinator peer the negotiation process starts. If the coordinator peer and the potential worker peers agree to work together, the execution of the task will start. From the coordinator's request to the delivery of the final result, each task goes through several stages.

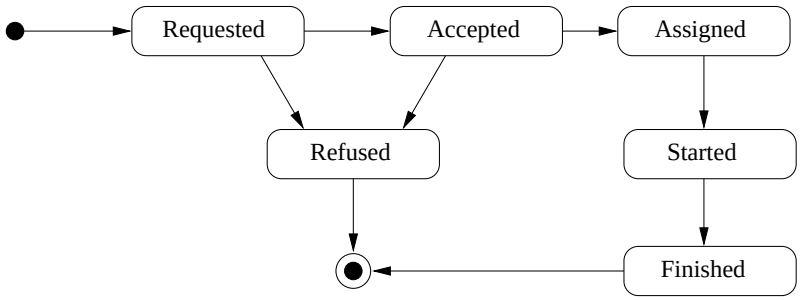

Figure 6.9: Task Status [Sch04]

As shown if Figure 6.9 the following transitions are possible:

- If a coordinator peer wants to execute a task, it has to send requests to the peers which are offering it. The task will then be marked as *REQUESTED*.
- The status will change to *ACCEPTED* if a worker peer agrees to perform the task.
- The worker peer can also refuse to execute a task. The task will then be marked as *REFUSED*.
- If the worker peer agreed, the coordinator peer assigns the task. It will be marked as *ASSIGNED*.
- The coordinator peer can also refuse to assign a task to a worker peer. The status changes to *REFUSED*.
- When all required input documents are available and all preceding tasks are finished, the execution of the task starts and it is marked as *STARTED*.
- Upon finishing the execution and delivering the output documents, the final state *FINISHED* is reached.

6.2.6 Implementation

The current implementation of the MOTION workflow component consists of the following parts:

WFCoordinator

The `WFCoordinator` class is the main class of the workflow component and provides an interface for the application layer. It sends messages through the MOTION middleware and provides access to the local database.

WFDatabase

The `WFDatabase` class is responsible for the management of users, processes, and instances. It provides methods to access objects in the local database.

WFTask, WFData and WFTaskQos

Processes and tasks are both stored as `WFTask` objects. If a `WFTask` object is no subtask of another one, it is a process, else a task. `WFData` acts as data container for the documents associated to a specific container. At the moment `WFTaskQoS` provides basic QoS attributes as `duration`, `price` and `delegate`, which is a flag if a task can be delegated to another peer.

WFUser

`WFUser` contains a list of user coordinates and a list of all tasks the user provides. Tasks are grouped by the specific communities.

WFInstance

`WFInstance` contains information about instances where the user is involved in, and where a process has been published and instantiated.

WFInstanceTask

For each task a user is involved in, a `WFInstanceTask` object is stored in the database.

WFEventHandler

`WFEventHandler` listens to subscriptions of a specific `WFInstanceTask`. If an artifact matches the subscription, the artifact is automatically downloaded and the corresponding `WFData` object updated.

6.3 Use Cases

We will now define our QoS-related use cases, which will be an extended version of the use cases of the current MOTION workflow component [Sch04]. The existing use cases will be shortly described. After that changes to existing use cases and additional use cases will be discussed in detail.

6.3.1 Process Management

The current MOTION workflow component involves the following use cases:

- Create community: Add a new community, all workflow system members are informed and can subscribe to it.
- Remove community: Remove a community which is no longer required from the system.
- Load process description: A new process description (in BPEL format) is uploaded to the system as artifact and can be downloaded by the users.
- Add process to community: To make a process visible, it has to be added to a community. Community members are informed and can download its description file.
- Create process instance: Create a process instance and a corresponding instance community for communication.
- Download process description: Download artifact with BPEL process specification file from the coordinator to the local repository.
- Provide task: A peer can provide a task which must be a part of a known process.
- Define QoS attributes: The peer states the QoS parameters on which it will provide the service.
- Subscribe to community: To receive messages from a specific community, a peer can subscribe to it.

For our QoS implementation we will make the following changes:

- Provide task: If a peer wants to provide a service, it also has to specify the QoS ontology which should be used for that task. After the QoS model is loaded, the peer can make changes such as add or remove criteria. For each criteria property the values are required by which the peer can provide the task.

6.3.2 Task Negotiation

Task negotiation includes:

- Search provided tasks: Query all peers of a task the coordinator peer needs. Peers which can provide the task reply with an announcement to the sender.
- Request task from peer: Look up in the database for peers which provide the specific task, compare the different candidates by their QoS attributes and send a task request to the chosen peer.
- Accept task: Decide if to accept a received task request. If yes, send an acceptance message to the coordinator.
- Refuse task: Both, worker and coordinator peer, can refuse a task.
- Assign task: If a worker peer has accepted a task request, the coordinator peer finally assigns him to definitely execute the task.

The following extension has to be made:

- Request task from peer: For each service provider candidate the peer can check the corresponding QoS values of passed task invocations.

6.3.3 Instance Execution

Instance execution involves the following use cases:

- Provide instance input data: Before the worker peers can execute the tasks, the coordinator peer has to provide the required input documents. The documents will be wrapped into artifacts and published to the system where they can be downloaded from.
- Check availability of task input data: Check whether input data for a specific task is available or not.
- Download input data: Download input documents to the local repository.
- Start task: Start the specified task, this includes that the task has been assigned to the peer and all required documents are downloaded.
- Execute task: Execute specified task, process input documents and create output documents.
- Create task output data: Make output documents available to the system. Wrap them into artifacts and publish it to the instance community.
- Check completeness of task output data: Ensure that all required output documents are available by querying the instance community.
- Finish task: Set status of the task as finished and inform coordinator peer about successful execution.
- Check availability of instance output data: The coordinator queries the instance community for the required output documents which should have been published by the worker peers.
- Retrieve instance output data: If documents have successfully been retrieved, download them to the local repository.
- Finish instance: When all peers have finished the task execution and all output documents were gathered, the coordinator peer sets the task to FINISHED.

The following changes will be made:

- Finish instance: After task completion the service requester will send the QoS feedback to the service provider. After receving this information, the service provider adds this data to his QoS archive of past task invocations.

6.4 QoS Implementation

The following sections describe the implementation of our QoS model which was specified in Section 5.

During the implementation phase it became apparent that the current implementation of the MOTION workflow component (see Section 6.2) was not cleanly layered. This caused problems with the MOTION messaging component. A total reenginering of the MOTION system and its workflow component was out of the scope of this thesis, therefore our QoS component was not completely implemented into the MOTION system. The following sections concentrate on a proof-of-concept implementation using a dummy class which emulates the Workflow system.

6.4.1 Used Technologies and Resources

The MOTION platform and also its workflow component were both implemented in Java [Inc]. Therefore our QoS component will also be implemented in Java. Additionally the following technologies and resources were used:

RDF

The Resource Description Framework (RDF) [MM04] is a language for representing information about resources in the World Wide Web. RDF is intended for situations in which this information needs to be processed by applications, rather than being only displayed to people. It provides a common framework for expressing this information so it can be exchanged between applications without loss of meaning. Since it is a common framework, application designers can leverage the availability of common RDF parsers and processing tools. The ability to exchange information between different applications means that the information may be made available to applications other than those for which it was originally created.

RDF Schema

RDF Schema [MM04, BG04] can be used to describe a RDF vocabulary. RDF Schema does not provide application-specific ontologies to describe specific things (like the QoS parameters in our example). Instead, it provides the facilities needed to describe such classes and properties, and to indicate which classes and properties are expected to be used together. In other words, RDF Schema provides a type system for RDF.

Jena

Jena [LP] is a Java framework for building Semantic Web applications. It provides methods to read, create, manipulate and write RDF and RDF Schema files.

JTreeTable

To enable an user-friendly interface for the ontology editor component, the `JTreeTable` [Mil06] layout component was used as a starting point and step-wise adapted to our QoS model in RDF. It is a combination of Java's build-in `JTree` and `JTable` layout components, which allows a clear presentation of hierarchical data such us our QoS ontologies.

6.4.2 Architecture

Figure 6.10 illustrates the class diagram of the `eu.motion.tuv.qos` package.

6.4.3 QoS Components

The basic components are build on the basis of the Model-View-Controller metaphor. The core of our QoS implementation consists of three classes: *QosLauncher*, *QosGUI*, and *QosCriteria*.

QosLauncher provides a dummy class where three buttons can be used to access the use cases where QoS criteria will be used (see Section 6.3). *QosGUI* provides the graphical user interface

Figure 6.10: Class Diagram of the eu.motion.tuv.qos Package

(GUI) to view and edit the specified QoS ontology. Finally, *QosCriteria* provides all necessary methods to load and manipulate QoS ontologies.

QosLauncher

The `QosLauncher` class is our dummy class which starts the QoS assistant prototype and emulates our three use cases. It provides (among others) the following methods:

main() The main method takes as argument the file path of our basic QoS ontology and starts our QoS Launcher application (see Figure 6.11). After a QoS ontology has been loaded, the file paths of this RDF file and its corresponding RDF data file (containing the data values) are also shown in the `QosLauncher` window.

provideTask() This method reacts on a click on the "Provide task" button. To provide a new task an existing RDF Schema file has to be selected. The selected QoS ontology is loaded and after that the `QosGUI` will be displayed. This enables the user to edit the specified QoS ontology and

Figure 6.11: QosLauncher Window

enter the default values, by which a task can be provided. These entered values will be written in a RDF data file, which resides in the same directory as the given RDF schema file.

viewTaskQos() After a RDF Schema file was specified, the given ontology can be viewed at any time of the workflow process. In our dummy `QoSLauncher` class the ontology can be viewed by a click on the "View task QoS" button. In this mode no alteration of the QoS ontology or the assigned values is possible. The user is only allowed to view the task's QoS ontology and attributes.

finishTask() When a task is finished, the service requester peer has finally to rate the consumed service. The corresponding editor window can be opened with a click on the "Finish task" button. In this case the user can only enter the values for each property, an alteration of the QoS ontology is not allowed.

QosGUI

The QosGUI class forms the graphical user interface (GUI) of our QoS ontology editor. It is based on the `JTreeTable` layout component (see Section 6.4.1).

There are three working modes for our GUI component:

MODE_EDIT_ONTOLOGY_AND_CHANGE_VALUES This working mode allows the user to edit the QoS ontology and to enter values for each QoS property in the ontology.

MODE_CHANGE_VALUES In this case the user is only allowed to enter values for each QoS property.

MODE_VIEW Finally, in this mode any alteration of the ontology or property values is prohibited. The user can only view the entered data.

These modes also reflect the architecture of the `QosGUI` class. The important methods in this class are:

QosGUI() The constructor method creates a new `QosGUI` object. It takes as arguments the working mode (see paragraph above), the default model file path (specified as command line parameter to QoSLauncher, see Section 6.4.3) and the file paths to the model file and data file.

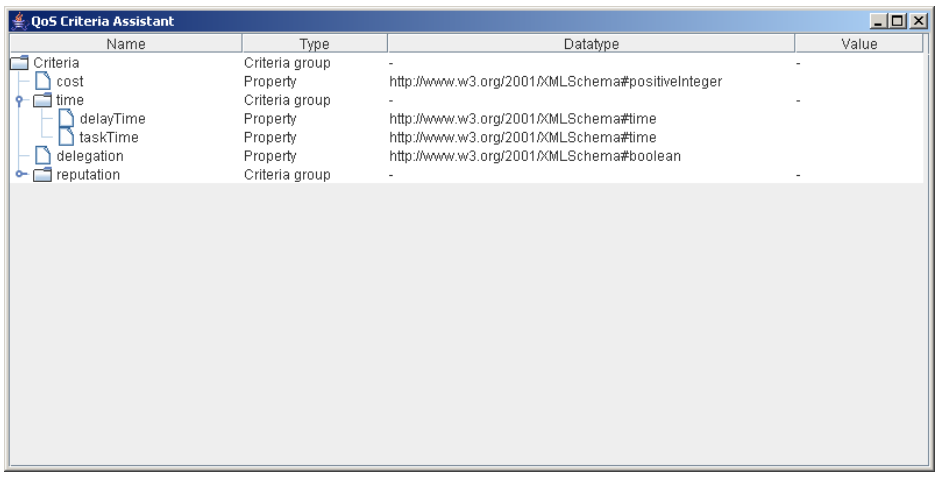

Figure 6.12: QosGUI Assistant Window

show() This method loads the layout components and displays the QoS assistant window for the specified QoS ontology (see Figure 6.12). Composite criteria groups can be folded out by clicking on the folder icons in front of each criteria group.

mousePressed(), mouseReleased() This two methods react on the user's actions. On the one hand, they display the popup context menu for the selected table row (see the showPopup() method). On the other hand, they invoke the necessary methods to alter the QoS ontology (provided by the `QosCriteria` class), after prompting for necessary information such as criteriaURIs, data types or input values.

By selecting a criteria row in the QoS assistant window and clicking on the right mouse button, the following context menu opens (see Figure 6.13):

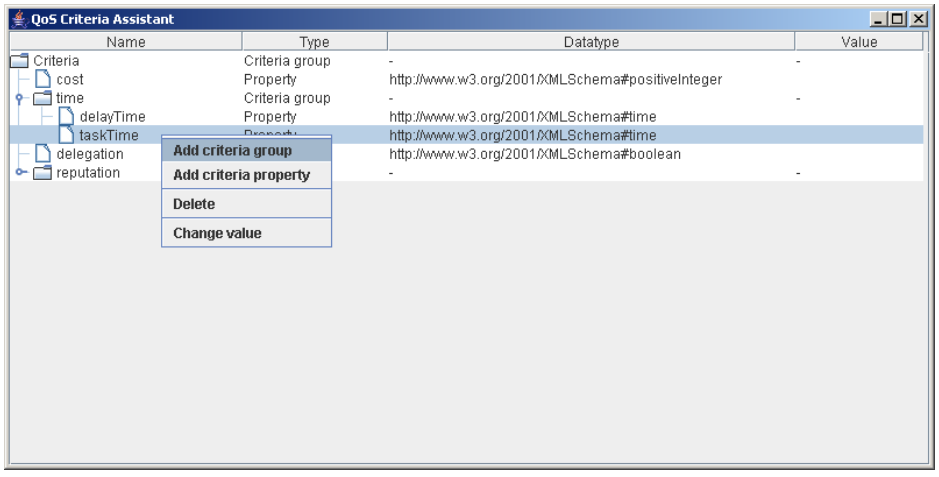

Figure 6.13: QosGUI Assistant Window, with Context Menu (Working Mode: Edit Ontology + Change Values)

It shows the following options:

Add criteria group Add a new criteria group to the QoS ontology. A prompt for the criteria group name will be shown.

Add criteria property Add a new criteria property to the selected criteria group. Two prompts for the property's name and its data type will be shown.

Delete Delete the selected criteria group or criteria property from our RDF ontology.

Change value Change the value of the selected criteria property. This option is only applicable to criteria properties, not to criteria groups.

The options shown in the context menu depend on the working mode, under which the QosGUI assistant window runs. The options above will be shown if the alteration of the QoS ontology and the entering of values is allowed. If the user is only allowed to enter values, then just the "Change value" option is shown (see Figure 6.14). If the GUI runs in the "view task QoS" mode, no context menu will be shown.

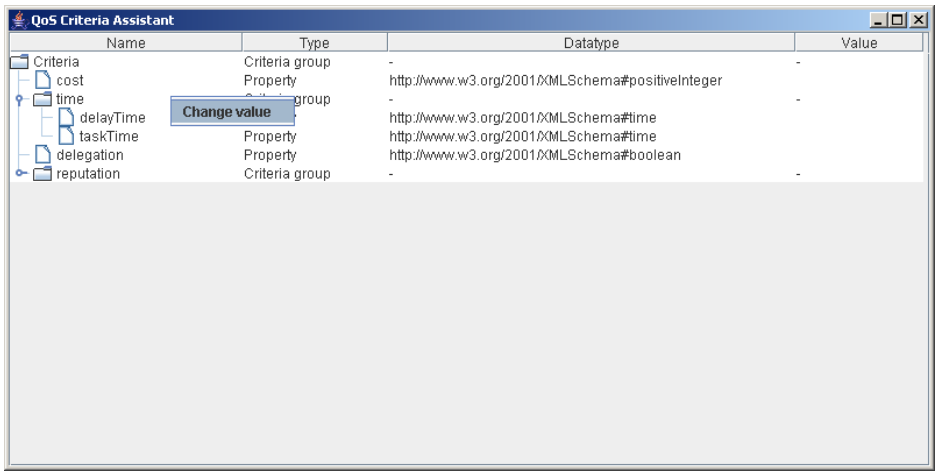

Figure 6.14: QosGUI Assistant Window, with Context Menu (Working Mode: Change Values)

QosCriteria

On the ground of our Model-View-Controller approach the `QosCriteria` class contains all necessary methods for the alteration of our QoS ontology.

QosCriteria() The constructor of the `QosCriteria` class takes the different file paths (default model file, model file and data file) as attributes.

add() By specifying the URI (Unique Resource Identifier) of the parent resource we can add new criteria to our RDF model. There are two different methods: One for adding a new criteria group (takes parentURI and criteriaURI as arguments) and the other for creating a new criteria property (needs parentURI, criteriaURI and the criteria's data type as arguments).

delete() Delete the specified criteria group or criteria property from our RDF model. A criteria group may only deleted, if there are no existing sub properties in this group anymore.

init() Init our QoS ontology model by loading the default model file.

load() Load the specified RDF and RDF Schema file into the given target model.

store() Store the current model tree to the specified output file. This method is called after each invocation of the add() or delete() methods.

6.4.4 JTreeTable Components

As stated before the user-friendly QoS assistant component is based on the `JTreeTable` component [Mil06]. It consists of the classes below:

ITreeTableModel

A new interface, extending the `TreeModel` interface, which describes the kind of data that can be drawn by a `TreeTable`.

AbstractTreeTableModel

A base class for TreeTableModels. This class handles the list of listeners.

TreeTableModelAdapter

A wrapper class that implements the `TableModel` interface, given both `TreeTableModel` and a `JTree`.

AbstractCellEditor

A base class for CellEditors. This class handlers the list of listeners.

JTreeTable

A subclass of `JTable`, this class can render data from a `TreeTableModel`.

RdfModel

A model of our QoS RDF model, implemented as a concrete subclass of `AbstractTreeTableModel`. This class implements the `ITreeTableModel` interface.

RdfModelNode

An inner class of `RdfModel` which recursively browses our RDF model for children and sub nodes.

TreeTableCellRenderer

The renderer used to display the tree nodes, a `JTree`.

TreeTableCellEditor

The editor used to interact with tree nodes, a `JTree`.

Chapter 7

Conclusion and Future Work

Specifying QoS criteria in the domain of Peer-To-Peer networks is still virgin soil. Since it is a totally new area where almost no reference work exists, a formal step-wise approach was chosen.

7.1 Summary

This thesis introduced step by step the theoretical backgrounds and a formal specification process (see Chapter 2 and Chapter 3). Furthermore, a detailed overview of the existing QoS approaches was given in Chapter 4. Out of this wide spread criteria, which was proposed in literature, the most important parameters were gained and specified as a basic QoS model in Chapter 5. This thesis showed an open, extensible framework for specifying quality of service criteria. The presented approach used today's state-of-the-art technologies like RDF and RDF Schema, which enable the user the extend existing models. Additionally, it is crucial that the chosen criteria is motivated by every-day service and not only technical service aspects. In Chapter 6 the underlaying MOTION Peer-to-Peer system and a proof-of-concept implementation of the proposed QoS model were described.

7.2 Result

The result of this thesis is an open and extensible framework for specifying service attributes and QoS parameters in a Peer-to-Peer environment. To produce meaningful and useful results, different groups of decision-makers have to be addressed. It is crucial that the specification of QoS attributes concentrates on the end-user perspective of a service, and not only on the technical aspects. The presented approach of QoS criteria classification and the proposed specification process can be used in any domain to formulate QoS criteria for a particular service.

The RDF (Resource Description Framework) language is the best choice for formulating such extensible ontologies. RDF is intended to formulate information which can also be processed by applications. In the domain of Web services there exist yet different approaches which propose standards for describing service interactions and service attributes.

Currently, it is unclear which technology for service interactions will prevail and become accepted. It is likely that an universal standard for QoS specification will be developed. However, with the increasing use of distributed networks the specification of QoS criteria of digital and non-digital services will become an important issue in the near future. It is only a matter of time.

7.3 Future Work

During the work the following topics turned out to be the potential area of future work:

7.3.1 Local Optimization and Global Planning

A powerful model for expressing and specifying QoS criteria builds the basis for further an optimization of tasks. The area of *Local Optimization and Global Planning* deals with the problem of optimization of task executions which can be conducted based on the specified QoS criteria. There are some existing approaches in the area of Web services for dynamic peer selection.

The domain of Peer-To-Peer networks has some extra characteristics which have to be considered: For example there is no global view of the entire system due to complexity issues and the assumption that peers can spontaneously depart from and reconnect to the system. That means that computation may only be done based on local information.

The Workflow system, in which all services are provided and executed, could also provide automatic measurement for some criteria properties. The RDF language provides for this purpose the necessary basis to exchange information between applications and to process this data automatically by other applications.

7.3.2 Security and Trust

The presented approach supposes that each user can view and edit all parameters. The real-world is usually different: Service are delegated to third-party service providers which do not need to know all information. In addition to that encryption and a kind of autorisation mechanism for the transfered data could become necessary.

Another area is trust. Who tells a service requester that the stated QoS values given by the service provider are accurate and representative? The approach of this thesis bypasses this problem by introducing a rating mechanism by which the service requester can finally judge the service after it is completed. For Peer-To-Peer networks a central database with the QoS values of all service providers is not practicable. It could be possible to query other peers for QoS information about a particalar peer, if they have consumed its service before. But this would increase network load.

Appendix A

RDF Representation

A.1 Basic QoS Model

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
   <!ENTITY qos "http://seal.ifi.unizh.ch/qos#">
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF
   xmlns="http://seal.ifi.unizh.ch/qos#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xml:base="http://seal.ifi.unizh.ch/qos#">
  <rdfs:Class rdf:ID="TimeCriteria">
    <rdfs:subClassOf>
      <rdfs:Class rdf:ID="Criteria"/>
    </rdfs:subClassOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="ReputationCriteria">
    <rdfs:subClassOf rdf:resource="&qos;Criteria"/>
  </rdfs:Class>
  <rdf:Property rdf:ID="cost">
    <rdfs:range rdf:resource="&xsd;float"/>
    <rdfs:domain rdf:resource="&qos;Criteria"/>
  </rdf:Property>
  <rdf:Property rdf:ID="delayTime">
    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
    <rdfs:domain rdf:resource="&qos;TimeCriteria"/>
  </rdf:Property>
  <rdf:Property rdf:ID="time">
```

```
30    <rdfs:domain rdf:resource="&qos;Criteria"/>
31    <rdfs:range rdf:resource="&qos;TimeCriteria"/>
32  </rdf:Property>
33  <rdf:Property rdf:ID="processFailures">
34    <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
35    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
36  </rdf:Property>
37  <rdf:Property rdf:ID="rating">
38    <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
39    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
40  </rdf:Property>
41  <rdf:Property rdf:ID="delegation">
42    <rdfs:domain rdf:resource="&qos;Criteria"/>
43    <rdfs:range rdf:resource="&xsd;boolean"/>
44  </rdf:Property>
45  <rdf:Property rdf:ID="taskTime">
46    <rdfs:domain rdf:resource="&qos;TimeCriteria"/>
47    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
48  </rdf:Property>
49  <rdf:Property rdf:ID="systemFailures">
50    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
51    <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
52  </rdf:Property>
53  <rdf:Property rdf:ID="reputation">
54    <rdfs:range rdf:resource="&qos;ReputationCriteria"/>
55    <rdfs:domain rdf:resource="&qos;Criteria"/>
56  </rdf:Property>
57 </rdf:RDF>
```

Listing A.1: RDF Schema Representation of the Basic QoS Model

A.2 Scenarios

A.2.1 Scenario A - Pizza Service

```
1 <?xml version="1.0"?>
2 <!DOCTYPE rdf:RDF [
3    <!ENTITY qos "http://seal.ifi.unizh.ch/qos#">
4    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
5    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
6 ]>
7 <rdf:RDF
8    xmlns="http://seal.ifi.unizh.ch/qos#"
9    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
10   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
```

```
11    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
12    xml:base="http://seal.ifi.unizh.ch/qos#">
13  <rdfs:Class rdf:ID="TimeCriteria">
14    <rdfs:subClassOf>
15      <rdfs:Class rdf:ID="Criteria"/>
16    </rdfs:subClassOf>
17  </rdfs:Class>
18  <rdfs:Class rdf:ID="PizzaToppings">
19    <rdfs:subClassOf rdf:resource="&qos;Criteria"/>
20  </rdfs:Class>
21  <rdfs:Class rdf:ID="ReputationCriteria">
22    <rdfs:subClassOf rdf:resource="&qos;Criteria"/>
23  </rdfs:Class>
24  <rdfs:Class rdf:ID="BillingCriteria">
25    <rdfs:subClassOf rdf:resource="&qos;Criteria"/>
26  </rdfs:Class>
27  <rdf:Property rdf:ID="pizzaToppings">
28    <rdfs:range rdf:resource="&qos;PizzaToppings"/>
29    <rdfs:domain rdf:resource="&qos;Criteria"/>
30  </rdf:Property>
31  <rdf:Property rdf:ID="deliveryTime">
32    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
33    <rdfs:domain rdf:resource="&qos;TimeCriteria"/>
34  </rdf:Property>
35  <rdf:Property rdf:ID="time">
36    <rdfs:domain rdf:resource="&qos;Criteria"/>
37    <rdfs:range rdf:resource="&qos;TimeCriteria"/>
38  </rdf:Property>
39  <rdf:Property rdf:ID="processFailures">
40    <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
41    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
42  </rdf:Property>
43  <rdf:Property rdf:ID="taskTime">
44    <rdfs:domain rdf:resource="&qos;TimeCriteria"/>
45    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
46  </rdf:Property>
47  <rdf:Property rdf:ID="pizzaRating">
48    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
49    <rdfs:domain rdf:resource="&qos;Criteria"/>
50  </rdf:Property>
51  <rdf:Property rdf:ID="cost">
52    <rdfs:range rdf:resource="&xsd;float"/>
53    <rdfs:domain rdf:resource="&qos;Criteria"/>
54  </rdf:Property>
55  <rdf:Property rdf:ID="delayTime">
```

```
    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
    <rdfs:domain rdf:resource="&qos;TimeCriteria"/>
  </rdf:Property>
  <rdf:Property rdf:ID="rating">
    <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
  </rdf:Property>
  <rdf:Property rdf:ID="delegation">
    <rdfs:domain rdf:resource="&qos;Criteria"/>
    <rdfs:range rdf:resource="&xsd;boolean"/>
  </rdf:Property>
  <rdf:Property rdf:ID="billing">
    <rdfs:domain rdf:resource="&qos;Criteria"/>
    <rdfs:range rdf:resource="&qos;BillingCriteria"/>
  </rdf:Property>
  <rdf:Property rdf:ID="systemFailures">
    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
    <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
  </rdf:Property>
  <rdf:Property rdf:ID="reputation">
    <rdfs:range rdf:resource="&qos;ReputationCriteria"/>
    <rdfs:domain rdf:resource="&qos;Criteria"/>
  </rdf:Property>
</rdf:RDF>
```

Listing A.2: RDF Schema Representation of Scenario A - Pizza Service

A.2.2 Scenario B - Online Music and Video Store

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
   <!ENTITY qos "http://seal.ifi.unizh.ch/qos#">
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF
   xmlns="http://seal.ifi.unizh.ch/qos#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xml:base="http://seal.ifi.unizh.ch/qos#">
  <rdfs:Class rdf:ID="TimeCriteria">
    <rdfs:subClassOf>
      <rdfs:Class rdf:ID="Criteria"/>
    </rdfs:subClassOf>
  </rdfs:Class>
```

```
18 <rdfs:Class rdf:ID="ProductsCriteria">
19   <rdfs:subClassOf rdf:resource="&qos;Criteria"/>
20 </rdfs:Class>
21 <rdfs:Class rdf:ID="MovieDimensionsCriteria">
22   <rdfs:subClassOf>
23     <rdfs:Class rdf:ID="MovieQualityCriteria"/>
24   </rdfs:subClassOf>
25 </rdfs:Class>
26 <rdfs:Class rdf:ID="ReputationCriteria">
27   <rdfs:subClassOf rdf:resource="&qos;Criteria"/>
28 </rdfs:Class>
29 <rdfs:Class rdf:ID="MovieQualityCriteria">
30   <rdfs:subClassOf rdf:resource="&qos;Criteria"/>
31 </rdfs:Class>
32 <rdfs:Class rdf:ID="SongQualityCriteria">
33   <rdfs:subClassOf rdf:resource="&qos;Criteria"/>
34 </rdfs:Class>
35 <rdf:Property rdf:ID="products">
36   <rdfs:range rdf:resource="&qos;ProductsCriteria"/>
37   <rdfs:domain rdf:resource="&qos;Criteria"/>
38 </rdf:Property>
39 <rdf:Property rdf:ID="time">
40   <rdfs:domain rdf:resource="&qos;Criteria"/>
41   <rdfs:range rdf:resource="&qos;TimeCriteria"/>
42 </rdf:Property>
43 <rdf:Property rdf:ID="processFailures">
44   <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
45   <rdfs:range rdf:resource="&xsd;positiveInteger"/>
46 </rdf:Property>
47 <rdf:Property rdf:ID="songQuality">
48   <rdfs:range rdf:resource="&qos;SongQualityCriteria"/>
49   <rdfs:domain rdf:resource="&qos;Criteria"/>
50 </rdf:Property>
51 <rdf:Property rdf:ID="taskTime">
52   <rdfs:domain rdf:resource="&qos;TimeCriteria"/>
53   <rdfs:range rdf:resource="&xsd;positiveInteger"/>
54 </rdf:Property>
55 <rdf:Property rdf:ID="productsNmbOfDownloads">
56   <rdfs:domain rdf:resource="&qos;ProductsCriteria"/>
57   <rdfs:range rdf:resource="&xsd;positiveInteger"/>
58 </rdf:Property>
59 <rdf:Property rdf:ID="movieWidth">
60   <rdfs:domain rdf:resource="&qos;MovieDimensionsCriteria"/>
61   <rdfs:range rdf:resource="&xsd;positiveInteger"/>
62 </rdf:Property>
```

```
63  <rdf:Property rdf:ID="cost">
64    <rdfs:range rdf:resource="&xsd;float"/>
65    <rdfs:domain rdf:resource="&qos;Criteria"/>
66  </rdf:Property>
67  <rdf:Property rdf:ID="productsNmbOfSongs">
68    <rdfs:domain rdf:resource="&qos;ProductsCriteria"/>
69    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
70  </rdf:Property>
71  <rdf:Property rdf:ID="delayTime">
72    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
73    <rdfs:domain rdf:resource="&qos;TimeCriteria"/>
74  </rdf:Property>
75  <rdf:Property rdf:ID="movieFrameRate">
76    <rdfs:domain rdf:resource="&qos;MovieQualityCriteria"/>
77    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
78  </rdf:Property>
79  <rdf:Property rdf:ID="rating">
80    <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
81    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
82  </rdf:Property>
83  <rdf:Property rdf:ID="delegation">
84    <rdfs:domain rdf:resource="&qos;Criteria"/>
85    <rdfs:range rdf:resource="&xsd;boolean"/>
86  </rdf:Property>
87  <rdf:Property rdf:ID="movieHeight">
88    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
89    <rdfs:domain rdf:resource="&qos;MovieDimensionsCriteria"/>
90  </rdf:Property>
91  <rdf:Property rdf:ID="movieDimensions">
92    <rdfs:range rdf:resource="&qos;MovieDimensionsCriteria"/>
93    <rdfs:domain rdf:resource="&qos;MovieQualityCriteria"/>
94  </rdf:Property>
95  <rdf:Property rdf:ID="productsNmbOfMovies">
96    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
97    <rdfs:domain rdf:resource="&qos;ProductsCriteria"/>
98  </rdf:Property>
99  <rdf:Property rdf:ID="systemFailures">
100   <rdfs:range rdf:resource="&xsd;positiveInteger"/>
101   <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
102 </rdf:Property>
103 <rdf:Property rdf:ID="songBitRate">
104   <rdfs:range rdf:resource="&xsd;positiveInteger"/>
105   <rdfs:domain rdf:resource="&qos;SongQualityCriteria"/>
106 </rdf:Property>
107 <rdf:Property rdf:ID="movieQuality">
```

```
    <rdfs:domain rdf:resource="&qos;Criteria"/>
    <rdfs:range rdf:resource="&qos;MovieQualityCriteria"/>
  </rdf:Property>
  <rdf:Property rdf:ID="reputation">
    <rdfs:range rdf:resource="&qos;ReputationCriteria"/>
    <rdfs:domain rdf:resource="&qos;Criteria"/>
  </rdf:Property>
</rdf:RDF>
```

Listing A.3: RDF Schema Representation of Scenario B - Online Music and Video Store

A.2.3 Scenario C - Supply-chain Management

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY qos "http://seal.ifi.unizh.ch/qos#">
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF
    xmlns="http://seal.ifi.unizh.ch/qos#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xml:base="http://seal.ifi.unizh.ch/qos#">
  <owl:Ontology rdf:about=""/>
  <rdfs:Class rdf:ID="TimeCriteria">
    <rdfs:subClassOf>
      <rdfs:Class rdf:ID="Criteria"/>
    </rdfs:subClassOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="ReputationCriteria">
    <rdfs:subClassOf rdf:resource="&qos;Criteria"/>
  </rdfs:Class>
  <rdf:Property rdf:ID="time">
    <rdfs:domain rdf:resource="&qos;Criteria"/>
    <rdfs:range rdf:resource="&qos;TimeCriteria"/>
  </rdf:Property>
  <rdf:Property rdf:ID="deliveryTime">
    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
    <rdfs:domain rdf:resource="&qos;TimeCriteria"/>
  </rdf:Property>
  <rdf:Property rdf:ID="processFailures">
    <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
  </rdf:Property>
```

```
<rdf:Property rdf:ID="productionFailureRate">
  <rdfs:domain rdf:resource="&qos;Criteria"/>
  <rdfs:range rdf:resource="&xsd;float"/>
</rdf:Property>
<rdf:Property rdf:ID="taskTime">
  <rdfs:domain rdf:resource="&qos;TimeCriteria"/>
  <rdfs:range rdf:resource="&xsd;positiveInteger"/>
</rdf:Property>
<rdf:Property rdf:ID="cost">
  <rdfs:range rdf:resource="&xsd;float"/>
  <rdfs:domain rdf:resource="&qos;Criteria"/>
</rdf:Property>
<rdf:Property rdf:ID="priorization">
  <rdfs:range rdf:resource="&xsd;boolean"/>
  <rdfs:domain rdf:resource="&qos;Criteria"/>
</rdf:Property>
<rdf:Property rdf:ID="penaltyRate">
  <rdfs:domain rdf:resource="&qos;Criteria"/>
  <rdfs:range rdf:resource="&xsd;positiveInteger"/>
</rdf:Property>
<rdf:Property rdf:ID="delayTime">
  <rdfs:range rdf:resource="&xsd;positiveInteger"/>
  <rdfs:domain rdf:resource="&qos;TimeCriteria"/>
</rdf:Property>
<rdf:Property rdf:ID="rating">
  <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
  <rdfs:range rdf:resource="&xsd;positiveInteger"/>
</rdf:Property>
<rdf:Property rdf:ID="delegation">
  <rdfs:domain rdf:resource="&qos;Criteria"/>
  <rdfs:range rdf:resource="&xsd;boolean"/>
</rdf:Property>
<rdf:Property rdf:ID="systemFailures">
  <rdfs:range rdf:resource="&xsd;positiveInteger"/>
  <rdfs:domain rdf:resource="&qos;ReputationCriteria"/>
</rdf:Property>
<rdf:Property rdf:ID="productionTime">
  <rdfs:domain rdf:resource="&qos;TimeCriteria"/>
  <rdfs:range rdf:resource="&xsd;positiveInteger"/>
</rdf:Property>
<rdf:Property rdf:ID="reputation">
  <rdfs:range rdf:resource="&qos;ReputationCriteria"/>
  <rdfs:domain rdf:resource="&qos;Criteria"/>
</rdf:Property>
```

```
78 </rdf:RDF>
```

Listing A.4: RDF Schema Representation of Scenario C - Supply-Chain Management

References

[BCS94] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: An Overview, 1994.

[BG04] Dan Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, February 2004. http://www.w3.org/TR/rdf-schema/.

[BHM+04] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. Web Services Architecture - What is a Web service? W3C Working Group Note, February 2004. http://www.w3.org/TR/ws-arch/.

[BM01] Paul V. Biron and Ashok Malhotra. XML Schema Part 2: Datatypes. W3C Recommendation, May 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

[BM04] Dave Beckett and Brian McBride. RDF/XML Syntax Specification (Revised). W3C Recommendation, February 2004. http://www.w3.org/TR/rdf-syntax-grammar/.

[CGea02] Francisco Cubera, Yaron Goland, and Johannes Klein et al. Business Process Execution Language for Web Services, August 2002. http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnbizspec/html/bpel1-0.asp.

[Chu91] Lawerence Chung. Representation and Utilization of Non-functional Requirements for Information System Design. In *CAiSE '91: Proceedings of the third international conference on Advanced information systems engineering*, pages 5–30, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[CP01] G. Cugola and G. Picco. PeerWare: Core Middleware Support for Peer-To-Peer and Mobile Systems, 2001.

[Cra98] E. Crawley. A Framework for QoS-Based Routing in the Internet. *IETF RCF 2386*, August 1998.

[CSM+04] Jorge Cardoso, Amit Sheth, John Miller, Jonathan Arnold, and Krys Kochut. Quality of Service for Workflows and Web Service Processes. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):281–308, April 2004.

[ea98] S. Blake et al. An Architecture for Differentiated Services. *Request for Comments (Informational) RFC 2475, Internet Engineering Task Force*, December 1998.

[Fen00] P. C. Fenkam. Dynamic User management System for Web Sites. Master's thesis, Graz University of Technology and Vienna University of Technology, Austria, 2000.

[FGRK01] P. Fenkam, H. Gall, G. Reif, and E. Kirda. A Dynamic and Customizable Access control System for Distributed Applications. Technical report, Distributed System Group, Technical University of Vienna, December 2001.

[GJS03] Janusz Gozdecki, Andrzej Jajszczyk, and Rafal Stankiewicz. Quality of Service in IP Networks. In *Communications Magazine, IEEE*, pages 153–159, Amsterdam, The Netherlands, September 2003. IEEE, IEEE Computer Society.

[GN02] X. Gu and K. Nahrstedt. A Scalable QoS-Aware Service Aggregation Model for Peer-to-Peer Computing Grids, 2002.

[Har01] William C. Hardy. *QoS: Measurement and Evaluation of Telecommunications Quality of Service*. Wiley, 2001.

[Inc] Sun Microsystems Inc. Java Plattform, Standard Edition (Java SE). http://java.sun.com/javase/index.jsp.

[ISO94] ISO 8402, Quality Management and Quality Assurance - Vocabulary, 1994.

[ITU] ITU-T Rec. E.800, Terms and Definitions Related to Quality of Service and Network Performance in Digital Networks, Including ISDNs, March 1993.

[ITU01] Support of IP-based Services Using IP Transfer Capabilities, ITU. March 2001.

[Ive62] Kenneth E. Iverson. *A Programming Language*. John Wiley and Sons, January 1962.

[KFRG02] Engin Kirda, Pascal Fenkam, Gerald Reif, and Harald Gall. A Service Architecture for Mobile Teamwork. In *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 513–518, New York, NY, USA, 2002. ACM Press.

[KS95] Narayanan Krishnakumar and Amit P. Sheth. Managing Heterogeneous Multisystem Tasks to Support Enterprise-Wide Operations. *Distributed and Parallel Databases*, 3(2):155–186, 1995.

[LNZ04] Yutu Liu, Anne H. Ngu, and Liang Z. Zeng. QoS Computation and Policing in Dynamic Web Service Selection. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 66–73, New York, NY, USA, 2004. ACM Press.

[LP] Hewlett-Packard Development Company LP. Jena – A Semantic Web Framework for Java. http://jena.sourceforge.net/.

[Mar94] S.P. Marsh. *Formalising Trust as Computational Concept*. Phd thesis, University of Sterling, Sterling, 1994.

[McC92] S. McCready. There is more Than One Kind of Workflow Software. *Computerworld*, 2:86–90, November 1992.

[Mil06] Philip Milne. Creating TreeTables in Swing, March 2006. http://java.sun.com/products/jfc/tsc/articles/treetable1/.

[MM04] Frank Manola and Eric Miller. RDF Primer. W3C Recommendation, February 2004. http://www.w3.org/TR/rdf-primer/.

[OEH02] Justin O'Sullivan, David Edmond, and Arthur Ter Hofstede. What's in a Service? Towards Accurate Description of Non-Functional Service Properties. *Distrib. Parallel Databases*, 12(2-3):117–133, 2002.

[PASW97] J. L. Abad Peiro, N. Asokan, M. Steiner, and M. Waidner. Designing a Generic Payment Service. *IBM Syst. J.*, 37(1):72–88, 1997.

[PZB88] A. Parasuraman, V.A. Zeithaml, and L.L. Berry. SERVQUAL: A Multiple-Item Scale for Measuring. Consumer Perceptions of Service Quality. *Journal of Retailing*, 64(1):12–40, 1988.

[Sch04] Daniel Schwarz. A Loosely-Coupled Peer-To-Peer Workflow System. Master's thesis, Technical University of Vienna, Austria, 2004.

[Tom06] Ioan Toma. D28.4v0.1 Non-functional properties in Web services. WSML Working Draft, March 2006. http://www.wsmo.org/TR/d28/d28.4/v0.1/.

[ZBN+04] Liangzhao Zeng, Boualem Benatallah, Anne H.H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327, 2004.