Specification of Quality Requirements for Business Collaborations

Bart Orriens

January 18, 2006

Abstract

Service-oriented computing (SOC) is the computing paradigm that utilizes services as fundamental elements for developing applications. In order to realize this vision quality is a critical issue that must be addressed. Current work in this area is either very abstract in nature or very low level; and without any connection between them. In this report we adopt a layered quality approach for specification of quality properties for business collaboration. The approach supports specification of high level quality objectives, the indicators to assess those, and the measures needed to calculate these indicators. Moreover, dependencies among objectives, indicators and measures are made explicit as such creating a traceable path from objectives to measures.

Infolab Technical Report Series, no. 26

January 2006

1 Introduction

Recently there has been increasing focus on service-oriented computing (SOC), the new emerging paradigm for distributed computing and e-business processing, to deliver flexible and adaptable corporate business services by utilizing existing services across organizational boundaries. *Business collaboration* refers to a cooperation between multiple enterprizes working together to achieve a common business goal. In order to realize the vision of utilizing services as fundamental elements for developing applications [14] for business collaboration, quality is a critical issue that must be addressed. Businesses will be averse to participating in cooperations that do not take place in an environment where delivered resources/services do not meet quality related requirements, for example concerning reliability, availability, efficiency and so on.

Therefore, for the successful adoption of SOC within the business collaboration domain, the paradigm must provide the means to make cooperation between enterprizes controllable from a quality point of view. At the moment, the most successful manifestation of SOC can be found in web services technology. A web service is a specific kind of service that can be unambiguously identified (generally by means of a URI) and whose service description and transport utilize open Internet standards, such as XML-based SOAP messages. Unfortunately, most work in the web service quality arena concentrates solely on defining low-level quality of service parameters; leaving the issue of relating these to higher level business requirements unaddressed. In this report we briefly introduce our business collaboration context framework, which provides the context required for business collaboration development and management. We also show how this context can be described using collaboration models. Subsequently, we explain how the introduced collaboration models can be augmented to support the specification of quality requirements from high level objectives to low-level quality measures such as provided by current web service quality solutions.

The remainder of this report is structured as followed: we first introduce a running example based on a complex insurance claim handling scenario in section 2. Next, in section 3 we briefly discuss our framework for business collaboration context; after which we explain our model driven approach for the definition of this context in section 4. After that, in section 5 we analyze the role of quality in business collaboration, and show how the specification of quality requirements can be facilitated. Finally, we present conclusions in 6 and outline future work.

2 Example

To exemplify the ideas presented throughout this paper an example inspired by the case study in [8] is used. The example describes a complex multi-party scenario, which outlines the manner in which a car damage claim is handled by an insurance company (AGFIL). AGFIL cooperates with several contract parties to provide a service level that enables efficient claim settlement. The parties involved are Europ Assist, Lee Consulting Services, Garages and Assessors. Europ Assist offers a 24-hour emergency call answering service to policyholders. Lee C.S. coordinates and manages the operation of the emergency service on a day-to-day level on behalf of AGFIL. Garages are responsible for car repair. Assessors conduct the physical inspections of damaged vehicles and agree repair upon figures with the garages. The scenario outline is as followed (more details are introduced in the remainder of this paper where needed):

The policyholder (customer) phones Europ Assist using a free-phone number to notify a new claim. The claim is received by a call handler within Europ Assist's telephone assistance department. After verification of the customer' credentials to ensure that the provided policy details are valid and the occurred loss is covered, the call handler finds an approved repairer nearest to the customer's location. The customer is notified that this repairer will arrive at the scene shortly, if necessary with a replacement car and towing service. The call handler subsequently contacts the selected repairer to notify him of the incident. If the repairer is not available, another one will be selected and contacted. The customer is kept posted of such changes by phone. Once the repairer is on its way, the call handler contacts AGFIL to inform them of the made claim.

Upon receipt of the claim a claim handler will be assigned within AGFIL. The claim handler will gather all related claim information like customer records, claim history, etc. to Lee C.S. After that the claim handler will fill out the claim details on a claim form, which is subsequently stored pending further developments. Lee C.S. in the meanwhile has one of its consultants working on the claim. The first thing this consultant does, is contact the garage to inquire about the status of the car. The garage has picked up the car while the previous was going on and has worked out an estimate of the car repair cost. If this cost was below $500 then the garage will have started repairs. But if the costs were higher, the consultant at Lee C.S. contacts an assessor to go to the garage and check out the car for him -or herself. This assessor makes an independent estimate of the repair costs and negotiates a final price with the garage.

The result of the assessment is next reported back to the consultant at

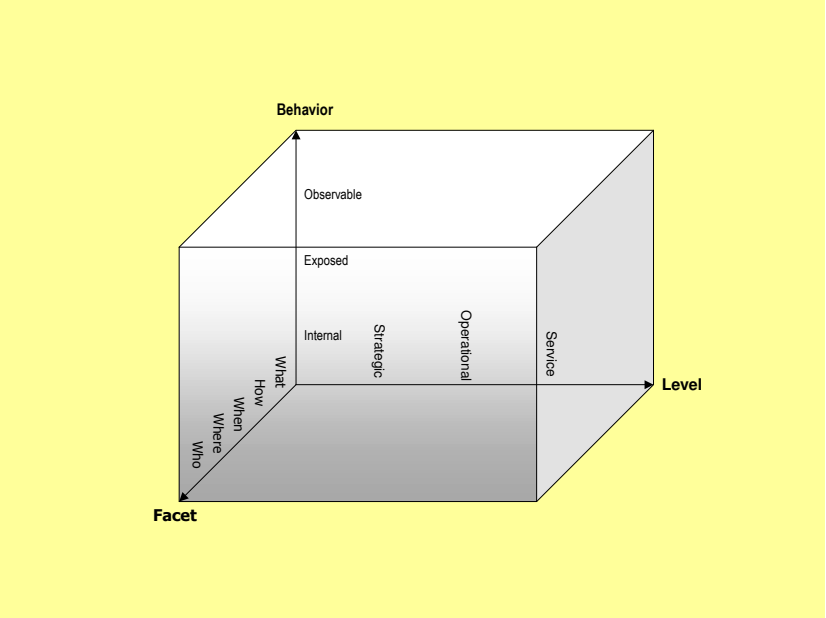

Fig. 1: Business Collaboration Context Framework (BCCF)

Lee C.S. The consultant reads the report and approves repair. An approval notification is sent to the garage, which consequently starts repairs on the car. Lee C.S' consultant also informs the claim handler at AGFIL of the final repair cost estimate upon which the claim handler incorporates the new information in the claim form. Once the garage has completed its repairs on the customer's car, an invoice is communicated to the consultant at Lee C.S. The consultant checks the invoice to see if it matches the earlier received cost estimate. Once the invoice is approved, the consultant sends the invoice onwards to AGFIL. The claim handler receives the invoice and adds it to the claim form. Payment for the claim is also issued.

3 Business Collaboration Context Framework

At the heart of our approach stands the Business Collaboration Context Framework (BCCF). The BCCF captures the context in which business collaboration development and management takes place by adopting a three dimensional view. Through this three dimensional view modularization of the definition and management of business collaborations is achieved. An overview of the framework is shown in Fig. 1.

As the figure illustrates we modularize the business collaboration context along three dimensions in the BCCF, being *behavior*, *level* and *facet*. We briefly discuss these in the following. For more information the reader is referred to [12, 11].

3.1 Behavior

The first dimension, **behavior**, places emphasis on the different behaviors that an enterprize exhibits in business collaboration; where consequently the purpose and target of development and management varies. The behavior dimension encompasses three types of behavior captured in three corresponding so-called collaboration aspects (inspired by among others [6, 15, 19]): observable, exposed and internal behavior expressed in the *conversation*, *participant public behavior* and *internal business process* aspect respectively.

The observable behavior constitutes the externally visible behavior between participants in a business collaboration; and is expressed in the *conversation aspect*. Captured in the *participant public behavior aspect* the exposed behavior describes how an individual participant can publicly behave in a business collaboration (i.e. its potential collaboration behavior). In contrast, the internal behavior (specified in the *internal business process aspect*) is also individual to each participant; however, it is only of interest to this particular participant, i.e. it can not be observed by other participants.

3.2 Level

The second dimension, **level**, recognizes the fact that the different business collaboration behaviors of an enterprize take place at several levels; where consequently the domain, degree of abstraction and the type of developers in development and management varies. In the BCCF three layers of abstraction are identified (inspired among others by [10, 20]): the *strategic*, *operational* and *service* level spanning from high level requirements to technical realization of collaboration behaviors.

At the strategic level the focus is on behavior that is abstract in nature, describing the purpose and high level requirements an enterprize has with the behavior. The operational conditions under which enterprizes exhibit their behavior are part of the operational level. This level establishes how high level strategic behavior (private, exposed and observable) will be operationalized. The technical realization of operational behavior is done at the service level, describing how the services provided by the IT-infrastructure support the operational activities.

3.3 Facet

The third dimension, **facet** captures the fact that the collaboration behaviors conducted by enterprizes affect many different parts. Facets represent these different parts of a business collaboration behavior that can be observed; and where consequently the focus and type of developer involved in collaboration development and management varies. Five facets are distinguished (inspired by among others [5, 16, 20]): *what*, *who*, *where*, *when* and *how* facet.

The *what* facet emphasizes the structural view of a collaboration behavior, focusing on what things are used to perform a collaboration behavior. The *how* facet takes a functional standpoint, and thus concentrates on how a collaboration behavior is conducted. The *who* facet concerns the participant(s) conducting the collaboration behavior. The location(s) at which the behavior is carried out are expressed in the *where* facet, whereas its temporal dimension is covered in the *when* facet.

4 Modeling the BCCF

To capture the three dimensions of collaborations aspects, levels and facets of BCCF we employ two types of model: meta models and models, both of which are defined for individual levels. Meta models provide design guidelines in terms of classes and their relationships, where depending on the collaboration aspect being modeled additional constraints are placed on the meta-model. Models represent a particular application design, and are derived by populating a meta model's *classes*.

Every meta model consists of six classes, where each class captures a particular facet; i.e. for *what*, *how*, *where*, *who*, *when* and *why* facet. Every class constitutes a set of logically related *attributes*. *Associations* connect the classes expressing dependencies among facets. *Mappings* define dependencies among levels by providing links between classes that describe the same facet at different perspectives (illustrated by the arrows between facets at different perspectives in Fig. 1).

Snippets of exemplary models for the AGFIL application are illustrated in Fig. 2, showing its strategic, operational and service model respectively; where the models are represented based on UML conventions. In order to distinguish different facets, we represent them in different shapes in their UML models (see also legend in Fig. 2): *what* facet is shown as folded corners, *how* facet as rounded rectangles, *who* facet as octagons, *where* facet as plaques, and *when* facet as hectagons. For more information the reader is referred to [12, 11]; where [13] contains the most recent details.

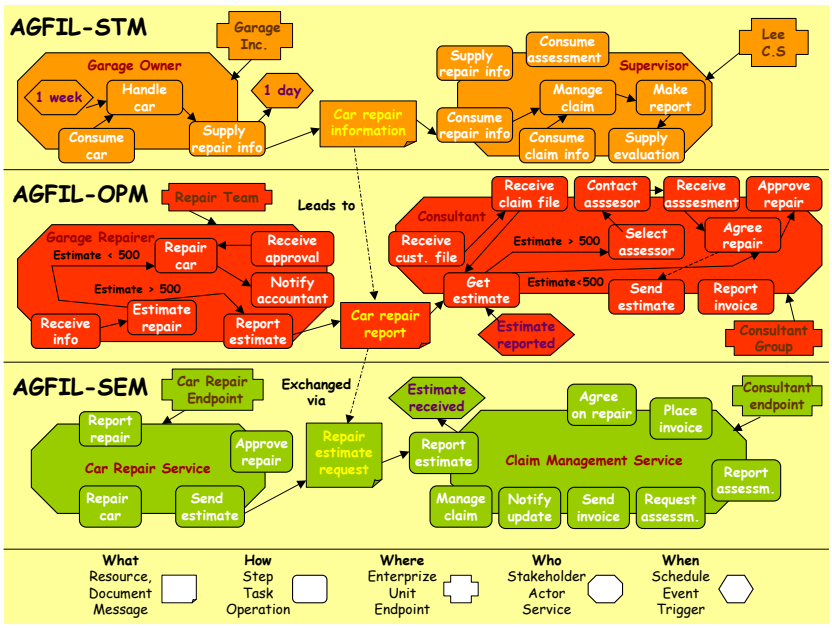

Fig. 2: AGFIL Collaboration Models

4.1 Strategic Models

At strategic level, strategic models like the AGFIL-STM in Fig. 2 capture purpose and high level requirements of business collaborations, akin to requirements analysis [2, 19]. Strategic models are expressed in terms of resources, steps, stake holders, enterprizes, and schedules. Resources such as `car repair information` provide abstractions for means such as financial, human and informational capital. Resources are used and produced by steps which represents high level functions.

Steps are of type 'internal' (like `handle car` presented inside the stakeholder boundary in Fig.2); or type 'communication' representing resource supply and consumption e.g. `consume repair information`. Stake holders like `garage owner` describe the participants involved who are responsible for carrying out defined steps. Stake holders belong to an enterprize, where enterprizes are manifestations record the information about the participating enterprize where behavior is carried out. Stake holders and their enterprizes are bound by schedules reflecting temporal constraints, like the deadline of `1 week` for `handle car`.

4.2 Operational Models

At operational level, operational models like the AGFIL-OPM in Fig. 2 depict how high level strategic behavior is realized in terms of operational activities. These are expressed in terms of documents, tasks, actors, units, and events. Documents (like `car repair report` represent the flow of information in a collaboration behavior. Documents are used and produced by tasks. Tasks represent specific business functions, and are of type 'internal' or 'communication' (represented inside or on the boundary of the actors respectively), e.g. `collect claim form` and `report invoice` respectively.

Actors such as `garage repairer` and `consultant` are responsible for carrying out tasks. Actors instantiate the **Actor** class and belong to units such as `repair team` unit, whose abstract definition is provided by the **Unit** class. In order to assess progress, keep logs to ensure non-repudiation, and etceteras, events are published and subscribed to by actors. Events describe business occurrences which have properties such as 'date', 'time', 'severity'.

4.3 Service Models

At service level, operational models are translated into service models that specify how the described operational behavior is realized using the services offered by the IT-infrastructure. Service models are defined in terms of messages, operations, services, endpoints, and triggers. Messages represents containers of information (e.g., `repair estimate request`), consisting of meta-data and actual data. Messages function as the inputs and outputs of operations such as `place invoice`.

Operations, just as steps and tasks at strategic and operational level respectively, can be dependent on one another. Additionally, they can be of type 'internal' or 'communication'. Operations are grouped in services (e.g. `car repair service`, which constitute collections of logically related operations. Services themselves are provided by endpoints (like `claim handling endpoint`) and have properties 'network location' and 'type'. To express technical occurrences triggers like `claim request acknowledged` can be defined on the basis of the **Trigger** class.

4.4 Mappings between Models

For the specification of dependencies between different collaboration behaviors at different levels, we employ vertical mappings. Vertical map-

pings are realized by providing links between the classes in different metamodels and instance models at different perspectives. The vertical mappings are based on the implicit links that exist between classes that describe the same facet at different levels in the same collaboration behavior. We define the following mappings:

Resources at strategic level are mapped to documents at operational level. Documents themselves are mapped to messages using *exchangedVia* relations. Steps are mapped via *decomposedIn* relations to tasks; whereas tasks are *realizedBy* operations. Stake holders *control* actors, where each actor is *representedBy* a service. Enterprizes are *organizedIn* units, where each unit itself *offers* one or more endpoints. Schedules are *splitInto* events, where each event *causes* multiple triggers.

5 Quality in Business Collaboration

Quality in general says something about "how good or bad something is" [3]. Interpreted in the context of business collaboration quality deals with providing assurance to enterprizes that the services they are exchanging meet their expectations. In other words, quality is concerned with providing peace of mind for the businesses involved, where they can rely on the fact that their collaboration can be relied upon from risks like unavailability of resources, slow responsiveness, and etceteras.

When put into the business collaboration context as presented in section 3, it follows that for business collaboration quality can be perceived at a strategic, operational and service levels, each of which represents a level of abstraction with its own content and meaning regarding quality. In the remainder of this section we shall discuss the role of quality and the specification of quality requirements at the different levels. An overview of the requirements that can be specified is provided in Figure 3; where this overview is based on an analysis of current literature such as the works in [1, 4, 7, 9, 17, 18, 21].

5.1 Strategic Level Quality

As observed at an abstract strategic level a business collaboration constitutes a cooperation between enterprizes making use of each other's business services to exchange resources to further their business goals. At this level quality specification deals with the definition of the objectives that the enterprizes have concerning quality with regard to the steps in these resource exchanges. Objectives analysis here is aimed at 1) identification

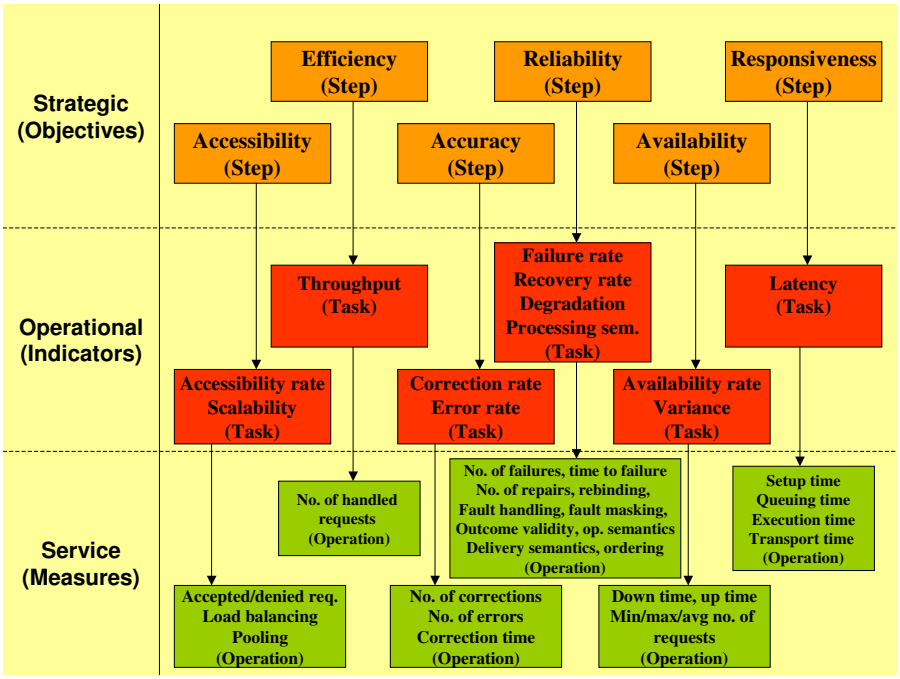

Fig. 3: Quality in Business Collaboration

of the objective, 2) measuring the magnitude of the potential loss if this objective is not realized, and 3) the probability that the objective will not be realized. For business collaboration we identify six types of quality objective:

- *Accessibility*

  Accessibility in general reflects the degree in which something is accessible; that is, whether it can be reached. In the context of business collaboration accessibility relates to whether or not a request for the performance of a particular step actually results in its performance. For example, `garage owner` may require that `consume repair information` is always accessible (i.e. it can be performed) between 09.00 to 17.00 on a weekday. Accessibility can be denied completely (no access), be limited or full.

- *Accuracy*

  The degree in which something is correct, is represented by its *accuracy*, i.e. whether something is without any mistakes. Here accuracy pertains to the correctness with which a step, such as `handle car` is performed. Depending on what is required accuracy can be set to 'sloppy', 'accurate' (i.e. normal) or 'highly accurate'.

- *Availability*

Whether or not something is available or not, depends on its *availability*. Available in general means that it can be immediately used; whereas in the context of business collaboration it reflects if a step can be conducted. Availability is a prerequisite for accessibility, that is, something is only accessible if it is available. Note though that the reverse is not true; e.g. `consume repair information` can be available but not accessible because the request is made after 17.00.

Availability can be high where a particular level is agreed, continuous with regard to operations (where a step can be performed at any time, but perhaps with some restrictions), and continuous availability; where a step can be performed at any time without any restrictions [1].

- *Efficiency*

  If something is done without any waste of time or resources, it is done efficiently. *Efficiency* captures this notion in business collaboration to indicate the degree in which a step is performed efficiently. If efficiency is important, it can be set to 'highly efficient' Otherwise, a setting of 'efficient' or even 'inefficient' may suffice; for example the case for `supply car repair information`.

- *Reliability*

  Reliability is a measure to determine the extent to which one can trust something to work as it is expected to work. In terms of business collaboration reliability expresses the degree in which one stake holder can trust another stake holder to perform a certain step as expected. Possible values for reliability are 'unreliable', 'reliable' and 'highly reliable'. For example, for `consume repair information` reliability will be highly reliable, as `repair information` is vital for the progression of the collaboration as a whole.

- *Responsiveness*

  The sixth and final objective is *responsiveness*, which represents the extent in which a response to a request is quick and positive. Responsiveness is also a characteristic of steps, and indicates how quick a step is carried out in response to the request made. Three levels of responsiveness are identified, being 'unresponsive', 'responsive' and 'very responsive'. `consume repair information`'s level might be set e.g. to 'responsive'.

Observe that the notion of 'performance' is often used in the literature to combine efficiency and responsiveness. Also, in addition to the above vari-

ous other quality objectives are sometimes specified (e.g. inter-operability or integrity in [18]); however, the six objectives described above are the most common ones. As such, we will focus on these objectives in the remainder of this report.

5.2 Operational Level Quality

Based on the objectives analysis at the strategic level quality indicators are defined for tasks at the operational level to establish a mechanism for assessing the identified objectives. At this level a business collaboration constitutes the sending and receiving of appropriate documents by enterprizes to further the state of the business collaboration. As such, the quality indicators that will be employed, help determine whether this document communication occurs in accordance with the quality objectives. In correspondence with the described objectives in subsection 4.1 we define the following quality indicators:

- *Accessibility* $\rightarrow$ *Accessibility rate, scalability*

  To be able to reason in a quantitative manner about accessibility, at operational level tasks are associated with an *accessibility rate* as well as *scalability* options. The former indicates the rate at which a task can be accessed; whereas the latter reflects the possibilities of scaling up and/or down to match actual demand. The accessibility rate thus functions as an indicator, where scalability indicates how accessibility can be facilitated (i.e. via up and/or down scaling).

- *Accuracy* $\rightarrow$ *Correction rate, error rate*

  The degree of accuracy of a step at strategic level is indicated by two rates at operational level for the tasks it is decomposed in. Firstly, *error rate* reflects how many errors occur; i.e. the absolute amount of mistakes in a given period of time. Secondly, *correction rate* takes the severity of an error into account by expressing how many errors can be corrected in a given period of time.

  Both rates are defined for individual tasks by specifying how many errors occur or are corrected respectively; where the time period is implicit in the period of time between the date/time of the event which started the task, and the date/time of the event signaling its completion. Naturally, the height of the error and correction rate is influenced by the degree of accuracy demanded form a strategic point of view; where each task must meet (or exceed) that accuracy level.

- *Availability → Availability rate, variance*

  To have a quantitative measurement for availability we employ the notions of *availability rate*, and *variance*. Availability rate quantifies when a step is available based on a 'continuous operations', 'agreed level' or a combination of the two. Depending on the availability scheme chosen at strategic level, the rate will apply regardless of a specific time period (determined by the date/time of a task's events). In the case of 'continuous operations' of a step, limited functionality will be reflected by a very low availability rate of one or more tasks (in which that step is decomposed).

- *Efficiency → Throughput*

  At operational level efficiency is expressed in terms of *throughput*. Throughput indicates how many times a task can be performed in a specific time period; where for each task decomposing a particular step it must be true that its throughput is at least high enough to meet the step's efficiency level as mandated on the strategic level (e.g. the throughput of `get estimate` mapped to `consume repair information`).

- *Reliability → Failure rate, recovery rate, degradation, processing semantics*

  Reliability falls into four indicators at operational level, being *failure rate*, *recovery rate*, *degradation* and *processing semantics* [7]. Failure rate represents the number of failures that are made in a certain time period; whereas recovery rate depicts the rate at with recovery from a failure is possible. Degradation reflects if failure is handled gracefully (e.g. according to some defined protocol) or abruptly. Finally, processing semantics indicate whether a task, in case of failure, is performed exactly once, at least once or at most once.

- *Responsiveness → Latency*

  The measurement indicating the responsiveness of a step at operational level is *latency*. Latency is the round-trip between the moment a task is started, and when it ends. The higher the level of responsiveness is specified at strategic level, the lower the amount of latency should be. As a step is decomposed into one or more tasks, it must be true for each task (like `report estimate` that the latency is below the threshold as mandated by the responsiveness level of this step (being `supply repair information`); since otherwise the average latency of the tasks combined (its sum divided by the number of tasks) will exceed the allowed latency level.

Together these indicators provide the measurements required to assess the achievement (or lack) of the strategic quality objectives. Furthermore, by linking their value to those of the corresponding quality objectives a traceability mechanism is established; which can be used to assess whether the objectives are achieved or not.

5.3 Service Level Quality

The quality indicators selected at the operational level must subsequently be calculated at the service level via quality measures. This level is the domain of the service oriented computing paradigm. In this paradigm a business collaboration is viewed as a set of interacting technical services, where these interactions are message based and facilitate the communication of information among services. In order to meet the business driven quality demands at this level, the qualitative properties of these message based interactions, i.e. message exchanges, must be adequately protected.

For this purpose we have identified the following quality measures, associated with individual operations; where these are discussed grouped in accordance with the quality indicator that they help measure:

- *Accessibility rate → No. of accepted requests, total no. of requests*

  The accessibility rate of a task at operational level is calculated at service level for each operation by dividing the *number of accepted requests* (i.e. handled messages) by the total number of requests made over a given period of time. Note that the average ratio of all operations combined must not drop below the task's accessibility rate; that is, if we sum of all operations its ratio of number of accepted requests/total number of requests, divided by the number of operations.

- *Scalability → Pooling, load balancing*

  To realize scalability (both up and down) at service level each operation can define the number of copies are kept in a pool ready to serve incoming requests. This number is captured in the *pooling* attribute. Similarly, via *load balancing* an operation can accommodate higher loads of requests; where it states the number of locations at which the same operation can be accessed.

- *Correction rate → Correction time, no. of corrections*

  The correction rate of a task is based on the rate of its mapped operations; where each ratio is calculated by taking the *number of corrections* divided by the total correction time; which represents the total

time it took to make the corrections. Naturally the sum of these ratios divided by the number of operations must equal or exceed the specified task correction rate. *Correction time* represents the time it on average takes to make one correction; where thus the total correction time should be roughly equal to the number of corrections made times the average correction time.

- *Availability rate → Downtime, uptime*

  The availability rate of a task at operational level is at service level calculated for each mapped operation by taking its *uptime*, and divide it by the sum of its uptime and *downtime*. The overall availability rate of the task is then the sum of ratios for all operations and divide it by the number of operations.

- *Variance → Minimum, maximum, average no. of requests*

  Variance is calculated for each operation using the standard statistical formula based on on the minimum, maximum and average number of request in a certain period. The variance of a task is then the sum of all operation variances divided by the total number of operations in which it is decomposed.

- *Throughput → No. of requests handled*

  At service level the throughput of a task is calculated per mapped operation by taking the *number of requests handled* divided by the elapsed period time. Overall throughput is then equal to the throughput of each operation, summing them, and divide them by the number of operations.

- *Failure rate → No. of failures, time to failure*

  The failure rate of a task is calculated using the *number of failures* per operation divided by a given time period; and then take the average of the ratios of these operations. Alternatively, *time to failure* is calculated to determine the time until a failure will occur.

- *Recovery rate → No. of repairs, repair time*

  The number of repairs made in a certain time frame is expressed in *number of repairs*; whereas *repair time* represents the time it takes to make a single repair. Naturally, the higher the average number of repairs to be made in a given period, the shorter the repair time should be.

- *Degradation → Fault handling, fault masking, rebinding, outcome validity*

Degradation of a task in case of failure is reflected in the task's realizing operations capability to do *fault handling*, *fault masking*, *rebinding* and *outcome validity*. If degradation is graceful, then one or more of these must be supported. Fault handling specifies how a fault is handled, being via halting execution, rolling back performed work, and/or terminating execution. Fault masking describes what faults an operation exposes to requesters; where an operation can hide failures, omissions, response failures, return value, incorrect state transition, timing failure (both late and early). Rebinding indicates whether in case of failure a requester has to rebind to the operation or not. Outcome validity reflects whether in case of failure the result of an operation is still valid or has become invalid.

- *Processing semantics → Delivery semantics, ordering*

  Lastly, processing semantics at operational level entail *delivery semantics* and *ordering* at service level. The delivery semantics for an operation can be exactly once, at least once or at most once. Ordering reflects whether messages must be send and received in order or not.

Observe that the above described measures at service level are not intended to be exhaustive in nature. The authors are aware that many other measures exist; the above is therefore intended to be of illustrative nature to show how operational quality indicators may be measured at a technical service level.

6 Conclusions

In this technical report we addressed the issue of specification of quality requirements for business collaborations. This work is motivated by the lack of support thereof in the current research with regard to relating high level quality objectives to concrete quality measures (excepting [7]); as most work (like [1, 9, 17, 18]) focuses on definition of low level quality measures without taking higher level, business driven quality requirements into consideration.

To remedy this situation we introduced our generic framework for capturing the business collaboration context; and explained how this context can be described via the usage of various meta models and models. After that we explained how these meta models and models can be augmented to facilitate quality requirement specification at strategic, operational and service level. Furthermore, we established relations between the requirements at

these different levels; as such enabling traceability of strategic quality objectives to operational quality indicators to service level quality measures (and vice versa).

A caveat concerns the defined quality properties: these are not intended to be exhaustive in nature nor do the authors expect them (and the relations between them) to be final. As the authors are not themselves experts in the field of quality, more work to further develop the (currently basic) support for quality requirement specification is required. However, we believe that the presented approach provides a first step on the road to comprehensive quality requirement specification for business collaboration.

References

[1] M. Bonnett, Understanding availability management: concepts and options, *White Paper, Systems Management Technical Support, IBM Corporation, Gaithersburg, USA, 2000*

[2] P. Bresciani et al, Tropos: An Agent-Oriented Software Development Methodology, *Autonomous Agents and Multi-Agent Sytems, Vol. 8, No. 3, pp. 203-236, 2004*

[3] Cambridge Learner's Dictionary, *http://dictionary.cambridge.org*

[4] J. Cardoso, A. Sheth, J. Miller, J. Arnold, K. Kochut, Quality of Service for Workflows and Web Service Processes, *Journal of Web Semantics, Elsevier, Amsterdam, The Netherlands, 2004*

[5] B. Curtis et al, Process Modeling, *Communications of the ACM, Vol. 35, No. 9, pp. 75-90, 1992*

[6] R. Dijkman et al, Service-oriented Design: A Multi-viewpoint Approach, *International Journal of Cooperative Information Systems, Vol. 13, No. 4, pp. 337-368, 2004*

[7] S. Froland, J. Koistinen, Quality-of-Service Specification in Distributed Object Systems, *Distributed System Engineering Journal, Vol. 5, No. 4, December 1998*

[8] P. Grefen et al, CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises, *International Journal of Computer Systems Science & Engineering, Vol. 15, No. 5, pp. 277-290, 2000*

[9] A. Mani, A. Nagarajan, Understanding quality of service for Web services, *http://www-106.ibm.com/developerworks/library/ws-quality.html?n-ws-1172, accessed January 12, 2006*

[10] Object Management Group, Model Driven Architecture, *http://www.omg.org/docs/ormsc/01-07-01.pdf, July 2001*

[11] B. Orriens et al, Bridging the Gap between Business and IT in Service Oriented Business Collaboration, *Proceedings of the IEEE International Conference on Services Computing, Orlando, Florida, USA, July 2005*

[12] B. Orriens et al, Establishing and Maintaining Compatibility in Service Oriented Business Collaboration, *Proceedings of the 7th International Conference on Electronic Commerce, Xi'an, China, August 2005*

[13] B. Orriens, Modeling The Business Collaboration Context, *INFOLAB Technical Report Series, No. 28, Tilburg, The Netherlands, January 2006*

[14] M. Papazoglou, G. Georgakapoulos, Introduction to the Special Issue about Service-Oriented Computing, *Communications of the ACM, Vol. 46, No. 10, pp. 24-29*

[15] C. Peltz, Web services orchestration: a review of emerging technologies, tools, and standards, *Hewlett Packard White Paper, January 2003*

[16] A. Scheer, Architecture for Integrated Information Systems - Foundations of Enterprise Modeling, *Springer-Verlag New York, Secaucus, NJ, USA, 1992*

[17] J. O'Sullivan, D. Edmond, A. ter Hofstede, What's in a Service? Towards Accurate Description of Non-Functional Service Properties, *Distributed and Parallel Databases, no. 12, pp. 117-133, 2002*

[18] R. Sumra, D. Arulazi, *http://www.developer.com/java/web/article.php/2248251, accessed January 12, 2006*

[19] P. Traverso et al, Supporting the Negotiation between Global and Local Business Requirements in Service Oriented Development, *Proceedings of the 2d International Conference on Service Oriented Computing, New York, USA, 2004*

[20] J.A. Zachman, A framework for information systems architecture, *IBM Systems Journal, Vol. 26, no. 3, pp. 276-292, 1987*

[21] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, H. Chang, QoS-Aware Middleware for Web Services Composition *IEEE Transactions on Software Engineering, Vol. 30, No. 5, May 2004*