A quality model for service monitoring and adaptation *

Cinzia Cappiello[1], Kyriakos Kritikos[1], Andreas Metzger[2], Michael Parkin[4], Barbara Pernici[1], Pierluigi Plebani[1], and Martin Treiber[3]

[1] Politecnico di Milano – Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci 32, 20133 Milano (Italy)
{kritikos, pernici, cappiell, plebani}@elet.polimi.it
[2] University of Duisburg-Essen – Software Systems Engineering
Duisburg-Essen, Germany
Andreas.Metzger@sse.uni-due.de
[3] Vienna University of Technology – Distributed Systems Group
Vienna, Austria
m.treiber@infosys.tuwien.ac.at
[4] Tilburg University – Department of Information Systems and Management
Tilburg, Netherlands
m.s.parkin@uvt.nl

Abstract.

1 Introduction

Based on the Service Oriented Architecture (SOA), Service Based Applications (SBAs) can be built from simple or complex services based on functional and non-functional requirements usually provided by a user. This construction of SBAs is usually performed either at design-time or run-time with the help of a service composition engine. As it is already known, the design-time construction of SBAs has the limitation that the constructed complex service can fail in many ways: one of its main component services may not be available as it has reached its capacity limit or it has produced erroneous output or it has failed or the network connecting this service with the outer world is not available. For the above reasons, the run-time construction of SBAs is preferred as it can solve the above problems, for instance, by substituting the faulty service with another one offering the same functional and similar quality of service (quality) capabilities with the faulty one. However, substituting a faulty service with a new one does not always solve the problem. Sometimes it is most preferable to re-execute the faulty service with the same or new input parameters or to compensate this service with a compensation action defined within the service management interface in case we are talking about transactional services. Moreover, in case

*The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube) and the Italian FIRB Project TEKNE

the faulty service is substituted with a new one, maybe the remaining execution plan has to be changed in order to still satisfy the user functional and quality requirements or violate the quality requirements in a smallest possible way.

Based on the above analysis, the highly dynamic environment under which services operate imposes new challenges for engineering and provisioning SBAs. SBAs have to be *flexible* and *adaptable*. By *flexible* we mean that the SBA should be able to change its behavior according to variable execution contexts while by adaptable we mean that the SBA should be able to execute even if the conditions at runtime differ from those assumed during the SBA's initial design. The former property can be achieved if SBAs are designed in such a way that are able to self-adapt to timely-respond to changes in their context or their constituent services or the user preferences and context. A necessary condition for achieving this property is that the SBA itself or another application can detect these changes and deliver them (in the second case) to the SBA. The latter property can be achieved by equipping the SBA with self-healing mechanisms that enable it to detect or even predict system failures and to react on them with adaptation actions that compensate for deviations in functionality or quality.

This paper focuses mainly on the quality aspect of SBA monitoring and adaptation. To this end, the first part of this paper deals with defining a quality model for SBAs. This quality model actually defines a complete set of quality attributes that can be quantified with specific metrics used to measure the quality capabilities of the SBA and consequently to detect if these capabilities deviate from the user specified quality requirements. While an SBA consists of three functional layers (from the highest to the lowest): the application, the service and the infrastructure layer, the set of quality attributes for the application and service layer is the same with the only difference that the quality attributes of the application level are derived from the quality attributes of the service level. For this reason and by considering also the fact that an SBA can be considered as a complex service, we define only two quality models: a *Service Quality* model and a *Quality of Infrastructure* (QoI) model. QoI can be used for configuring services and infrastructural components and influences significantly quality. So both quality models are equally important and must be used in SBA monitoring and adaptation. Especially, while QoI has been partially used in SBA monitoring, it has been neglected in SBA adaptation which is currently performed only at the service level. For this reason, we believe that by defining a QoI model we make a first step for a more complete quality-based adaptation of SBAs.

Quality violations are usually detected by monitoring the SBA, which offers a reactive way of adapting to these violations. In this paper, after explaining the main drawbacks of reactive adaptation, it is proposed that a proactive approach should be followed by predicting the future quality of the SBA and thus adapting the SBA before the actual deviation takes place. Based on the quality model proposed in the first part of the paper, it is advocated that quality attributes should be distinguished between those that can be measured objectively and those that can be measured subjectively. The quality of the former subset of quality attributes can be predicted by exploiting two traditional quality assur-

ance techniques: *testing* and *model analysis*. On the other hand, the quality of the latter subset of quality attributes can be predicted by using *reputation* and *rating* systems approaches. All of these prediction techniques are analyzed in detail and their advantages and disadvantages are highlighted.

The last part of this paper envisions adaptation as a self-healing behavior of a SBA and so adaptation is defined as a general mechanism for allowing a SBA to react proactively or reactively through one or more adaptation actions. Then the main aspects of adaptation are shortly analyzed which influence the decisions taken for reacting. Next, two main types of adaptation actions for quality are defined: *negotiation* and *repair*. The former allows the definition or redefinition of the quality attributes (and their thresholds) of interacting services and infrastructure components at design time or more dynamically following an auction-based approach. The latter is used to repair system errors with repair actions on both the service and infrastructural layer. Finally, it is argued that decision mechanisms for applying repair actions have only been examined at the service level and that the complexity of such decisions might require completely new approaches in future research.

2 Related Work

QoS research gained a lot of attention during the last years in the field of Service Oriented Computing (SoC). In particular service discovery was investigated with regard to QoS and the use of QoS attributes during the discovery process. The work presented in [1] proposes a quality extension to UDDI that encapsulates QoS information which distinguishes four basic classes of QoS attributes, namely *runtime related* QoS, *transaction support related* QoS, *configuration management and cost related* QoS and *security related* QoS.

The approaches presented in [2] and in [3] extend OWL-S ontologies to model QoS related information.

The work presented in [4] discuss a basic set of QoS attributes. These include *availability*, *accessibility*, *integrity*, *performance*, *reliability*, *regulatory*, and *security*. However, the authors only refer to this list and but do not provide a dedicated model for the attributes they introduce.

The work presented in [5] introduces five major quality criteria for atomic services: *execution price*, *execution duration*, *reputation*, *reliability*, and *availability*. The authors use these criteria in a linear programming approach to select optimal execution plans for composite services.

Sabata et. al. [12] present a taxonomy for the specification of QoS in distributed systems. In their approach, the taxonomy is a hierarchical structure that is divided into two major classifications: *metrics* and *policies*. Metrics such as performance (divided into *timeliness*, *precision*, and *accuracy*) measure quantifiable QoS attributes. Policies provide strategies to cope with changing situations and define renegotiation strategies, etc.

The work presented in [10] introduces a framework for monitoring Grid services with respect to QoS attributes. The authors define an extensive, hierarchi-

cal QoS classification schema that consists of four main categories, namely *cost*, *dependability*, *configuration* and *performance*.

3 Quality Modeling

3.1 Quality of Services

Previous service quality models and taxonomies considered a small number of quality categories and in each category only some of the representative quality attributes were contained. In our approach, this is changed. First of all, we do not consider only *domain-independent* quality categories and attributes but also some of the most frequent *domain-dependent* ones like the quality categories of *Data-related*, used for services operating on and/or producing data, and *Quality of Use Context*, used for context-aware adaptive services. Secondly, we consider quality categories and attributes that are relevant not only for the service and its service provider but also for the service requestor. For example, the *dependability* quality category is important for the service provider but not for the requestor while the *usability* quality category is important only for the requestor/user. Thus, we take into account both the *service provider* and *service requestor views*. Finally, in each category there is an extensive list of the most representative quality attributes including not only *atomic* but also *composite* quality attributes produced from atomic ones like *response time*, *failure semantics* and *robustness*.

In the sequel, we are going to shortly analyze our service quality model by focusing on each quality category in order to justify why we have included it, explain what is its purpose and describe some of its representative quality attributes. The graphical representation of our service quality model can be seen in Figure 1. Finally, we conclude this subsection by drawing directions for further research.

Performance The *Performance* quality category contains quality attributes that characterize how well a service performs. Two quality attributes with a very well defined meaning are common among all research approaches: *response time* and *throughput*. In our quality model, *response time* is regarded as a composite quality attribute computed from *latency* and *network delay*. Similarly, *latency* is composite and is computed from *execution time* and *queue delay time*. Finally, a quality attribute that has similar meaning with *execution time* is *transaction time* but is used in a different context (transactional services).

Dependability *Dependability* of a computing system is the ability to deliver service that can justifiably be trusted [6]. In the work of Avizienis et. al. [6], the phrase "justifiably trusted" is translated into three different quality attributes and views: *availability*, *reliability* and *security*. In our opinion, *security* is orthogonal to *dependability* and must be put in separate category because it provides the mechanisms that can possibly avoid a specific type of failures from happening but it has nothing to do with the way the service has been designed and built (with respect to its proper functioning). Moreover, security mechanisms can be broken so even these faults cannot be prevented. Thus, we believe that dependability contains *availability*, *reliability*, *failure semantics* [7] and *robustness* [1,

8, 9] with the latter two attributes describing: a) the type of faults that can be exposed and how the service reacts to them (the first one) and b) the capability of the service to behave in an acceptable way when these faults happen (the second one). Another important remark is that besides *availability*, another quality attribute with similar meaning that should be added in this quality group is *accessibility* [10] as it can characterize the case where a service is available but not accessible to some users e.g. due to network connection problems.

Security Services should be provided with the required *security*. With the increase in the use of services which are delivered over the public Internet, there is a growing concern about security. The service provider may apply different approaches and levels of providing security policy depending on the service requestor. Security for services [11, 1, 9] means providing *authentication*, *authorization*, *confidentiality*, *traceability/auditability*, *accountability*, *data encryption*, and *non-repudiation*. Besides these classical quality attributes, we have added two more, namely *safety* and *integrity* [6].

Data-related In specific application domains, services do not only accept input parameters but also input data and they may also produce output data. For example, a *credit card* service can accept as input a data file describing the user's credit card information and can produce as output a data file describing details of the transaction executed based on the functionality of the service. These input/output data are characterized by quality attributes that have been traditionally used in the information and data quality domains like *accuracy* and *timeliness* [8]. Except from traditional data quality attributes, we have added two more attributes that characterize the way the service behaves with respect to the data it operates on or produces when it fails (*data policy*) and the degree of validity of the data (*data integrity* [8]).

Configuration Management This quality group/category contains quality attributes that influence the way a service is configured to function (*service level* [12]) or characterize if the promised functional and quality level has been actually delivered during the service's lifetime period (*completeness*, *stability*, *reputation*).

Network and Infrastructure-related The network is usually used for sending requests and receiving (either instantaneously or continuously) the results back and connects the service with the requesting user. Initially, most of the research approaches [12, 4, 11, 1] were neglecting this quality aspect but after the work published in [13], this situation has changed [8, 9]. Network parameters influence the values of service quality parameters of other quality groups like *response time* and *availability*. We have identified four network quality parameters, that are common among all research approaches, namely: *bandwidth*, *network delay*, *delay variation* and *packet loss*. Another entity that is different from the service or the user but it influences directly or indirectly service quality is the infrastructure. This entity characterizes the service execution environment and can be characterized by many quality attributes. For the time-being, we have only identified three of them, namely: *server failure*, *guaranteed messaging requirements* and *security level*.

Usability Usability collects all those quality attributes that can be measured subjectively according to user feedback. It refers to the ease with which a user can learn to operate, prepare input for, and interpret the output of the service. This quality group contains three composite (that can be further decomposed) and two atomic quality attributes. The definition of these attributes is given in the table.

Quality of Use Context Services can become adaptive if they can change their configuration, behavior and appearance based on the context, where "context is any information than can characterize the situation of the entity. An entity is a person, place or object that is considered relevant to the interaction of a user and an application including the user and the application themselves" [14]. So based on this definition, which is quite general, context is any information that characterizes the service and its user, their physical and execution environments (including the devices used) and the network that connects them. Context information has also quality [15–17] as it depends on the way it is sensed or derived, the time that it is produced and delivered, the level of detail and other factors. Thus, adaptive services should be designed and executed taking also into account the quality of the context that is delivered to them so as to be able to make rational and realistic decisions when to adapt and how. After reviewing the related literature in quality of context, we have identified seven (7) quality attributes from which the most important ones are: *precision* (how precise is the information), *resolution* (the level of detail), *probability of correctness* and *freshness* (age of the information).

Cost Some research approaches consider cost as a service attribute that is orthogonal to the service quality because it is related to both functional and non-functional service attributes. However, the majority of research approaches [1, 18, 10, 8, 9] considers cost as a service quality attribute. In addition, all research approaches, at least the ones we have studied, use cost at the service selection phase in order to select the best service according to its QoS and cost and user's preferences and budget. Based on the above reasons, we regard cost as a (composite) quality attribute (and group) consisting of three (atomic) service attributes: *cost model*, *fixed costs* and *variable costs*. Actually, cost can be computed either from all atomic cost attributes or only from the *fixed costs* attribute.

Other This quality category has been created to contain various quality attributes of services that do not belong to any other category. So the contained quality attributes may not be related to each other. For the time being, only one quality attribute has been considered called *supported standards* [4, 11, 1, 10, 8, 9] used to indicate if the service complies with standards or not. This attribute can affect the portability of the service and its inter-operability with other services or applications.

A quality model of a service (and its infrastructure) is the first step for defining service quality. The second and more difficult step is to associate the quality categories and attributes with each other modeling in this way their quantitative and qualitative dependencies. This step is essential if we want to be

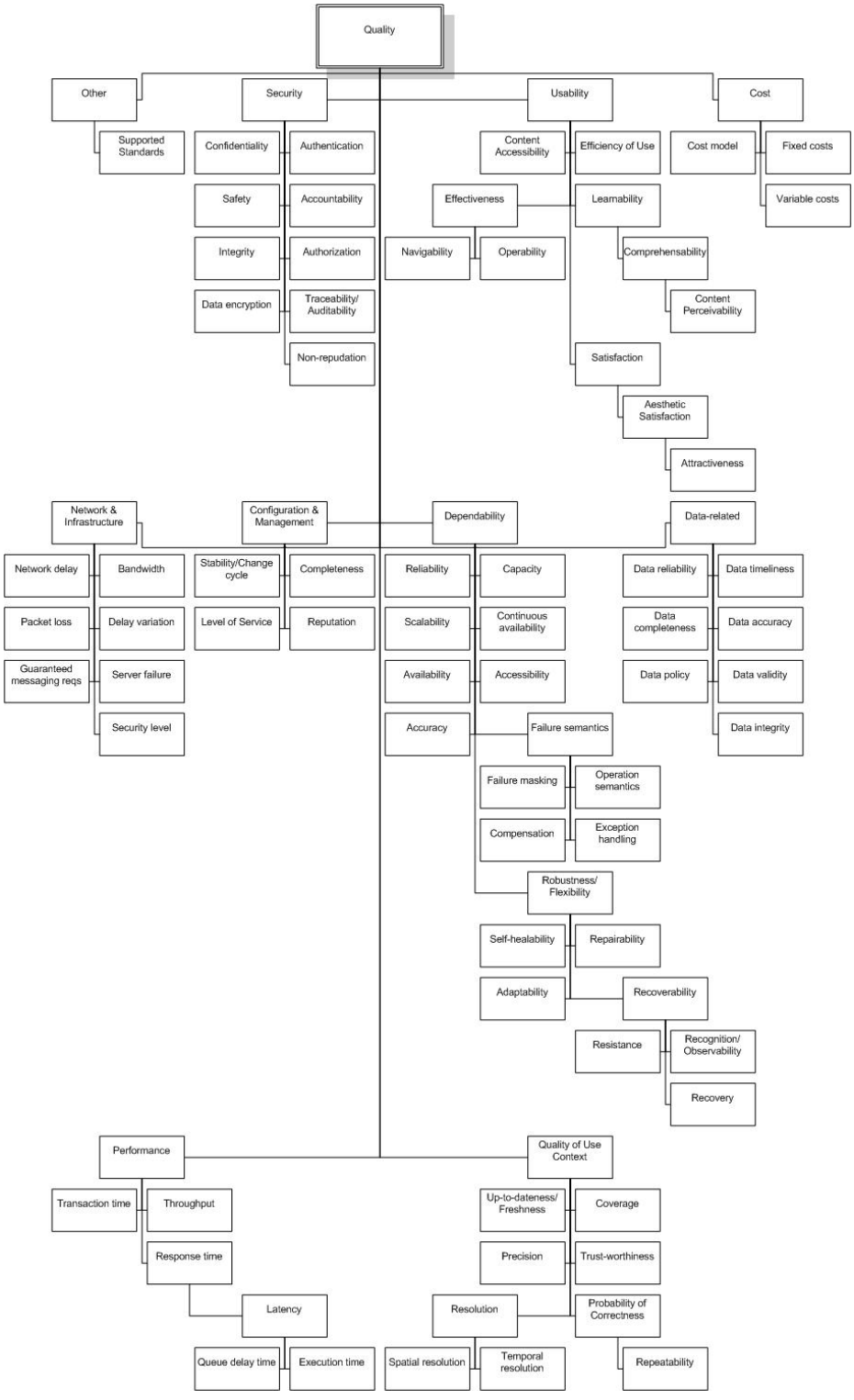

Fig. 1. Service Quality Model

able to derive more information from measurements or to evaluate the correctness of these measurements or of quality predictions. A further step will be a definition of a semantic, rich and extensible quality metamodel that will include all possible concepts and their relationships and inter-dependencies required for defining and monitoring end-to-end quality characteristics and negotiating SLAs.

3.2 Quality of infrastructure

4 Quality prediction

5 Quality Prediction

Service-based applications (SBAs) operate in highly dynamic and flexible contexts. Those applications should therefore be able to self-adapt to timely respond to changes in their context or their constituent services, as well as to compensate for deviations in functionality or quality. Currently, such a self-adaptation often happens after a change or a deviation has occurred. Yet, such reactive adaptations have the following drawbacks (cf. [19]):

- Executing faulty services can lead to unsatisfied users, can result in loss of money (e.g., wrong bank transactions) and typically requires the execution of additional activities (for instance, compensation actions must be planned / designed).
- Execution of adaptation activities takes time and thereby can reduce the system performance (e.g., response to user input).
- It can take time before problems in the system lead to monitoring events (e.g., time needed for the propagation of events from the infrastructure to the business process level), thus events might arrive so late that an adaptation of the system is not possible anymore.

As a consequence, SBAs should be able to proactively adapt to prevent the above drawbacks. Key to proactive adaptation is to predict the future quality (and functionality) of a SBA and to proactively respond if the prediction uncovers deviations from expected quality (or functionality).

The previous section has introduced a taxonomy of quality attributes, which are relevant for SBAs. In this section, we propose different kinds of approaches in order to predict the different types of quality attributes. We envision two major classes of approaches, based on two key types of quality attributes for SBAs and services (e.g., see [**?**]):

- *Quality of Service (QoS) attributes:* QoS attributes can be measured objectively and are the typical constituents of Service Level Agreements (SLAs [**?**]). Examples for QoS attributes are performance, availability, ((to be completed based on taxanomy))

– *Quality of Experience (QoE) attributes:* QoS attributes can have a subjective element to them and typically reflect the perception of individual or groups of service users. Examples for QoE attributes are usability, trust, ((to be completed based on taxanomy))

For what concerns QoS attributes, more traditional approaches can be employed in order to predict the future quality of a service-based application. Section 5.1 sketches two ideas on how existing quality assurance techniques can be exploited. Moreover, this section also analyzes another idea of using benchmarks to predict the QoS of a service.

For what concerns QoE attributes, the advent of the Web 2.0 [21] provides novel tools and platforms, which can be exploited. Examples include reputation systems (used in platforms like YouTube, Flickr, or eBay), as well as social bookmarking tools (like Delicious, Digg and StumbleUpon). Those tools and platforms open the door for novel ways of determining and predicting QoE attributes. This is sketched in Section 5.2.

5.1 Predicting Quality of Service Attributes

Quality prediction based on testing: Testing can be considered as a means to measure / assess the quality of a system. As an example, through performance tests, the systems response time under different loads can be measured. In order to extend the traditional testing techniques towards quality prediction, one idea is to use online testing techniques, i.e. to perform testing activities in parallel to the operation phase of service-based applications (in contrast to offline testing which is done during the design phase). Obviously, an online test can fail; e.g., because a faulty service instance has been invoked during the test. This reveals a potential problem that the service-based application can face in the future of its operation; e.g., when the application invokes the faulty service instance. In [Hielscher et al. 2008] an initial framework for such a use of online testing has been presented.

Quality prediction based on model analysis: Another approach which allows predicting the future quality of a system is to exploit models to reason on QoS attributes during the run-time of an SBA. As an example, the approach presented in [20] uses models, which are continuously updated during the operation of the SBA, in order to predict violations in the future execution (states) of the system.

Quality prediction based on benchmarks Another possible strategy is to use benchmarks as means to calculate expected performance of services in new environments. As prerequisite server side benchmarks can serve as baseline to establish a metric for the performance of the service environment. Note that these numbers only provide a *rule of thumb* and provide an rough estimation for the expected performance. In this context, one must distinguish between (i) data centric services that depend on databases (e.g., the provision of business reports) and (ii) services that perform data transformations (e.g., transformation of ASCII text to PDF, etc.) or calculations (e.g., scoring values of companies based on balance sheet data). In the case of data centric services, the major

part of the performance depends on the performance of the database. Database benchmarks [5] can be used to establish a performance index for these type of services. Services that do not use databases but do data transformation depend on criteria that can be measured by benchmarks that focus on CPU, memory and I/O performance [6].

5.2 Predicting Quality of Experience Attributes

Quality prediction exploiting Web 2.0 technologies: The Internet is currently evolving towards the "Web 2.0", in which social networks, folksonomies, reputation and rating systems play an ever more dominant role [21]. Reputation systems attempt to rate entities (e.g., book, images, videos, etc.) based on a collection of opinions (subjective perception). As examples, Flickr provides individual (subjective) opinions of people about certain photos, the videos provided by YouTube are rated by the viewers of these videos, the participants in eBay are rated based on the experience of previous sales.

Considering the fact that in the future Internet of Services, an abundance of services will be available and accessible over the Web, it is only natural to assume that a rating system for these services will be put in place (cf. [**?**]). We believe that this fact can be exploited to measure and predict QoE attributes, i.e. to determine quality attributes that are strongly influenced by how the quality of a service has been perceived by the users of that service.

In this respect, we envision exploiting techniques for aggregating individual views into a public opinion (as has been done for the quality attributes trust and reputation; e.g., see [22, 23]). As an example, this approach could be extended and adapted to aggregate the individual user experience about the usability of a service. Analyzing the change of the public opinion over time, could then support predicting the future quality.

6 Adaptation

The quality model and quality prediction allow the assessment of services and they are the basis for enforcing actions on the system to guarantee that users' requirements are satisfied.

In general such actions have the goal of guaranteeing the functionalities and quality of the system even when perturbed situations arise. In general, a self-healing behavior is envisioned. Prediction allows acting before actual failures occur in the system (*proactive approach*), while a *reactive approach* based on occurred failures is adopted after failures. In the following, we define adaptation as the general mechanism which allows reacting either in a proactive or a reactive way, as a combination of one or more adaptation actions.

[5] http://www.tpc.org/
[6] http://www.specbench.org/

Several aspects have to be considered for adaptation, which are based on implicit or explicit classification of error states in the system. We illustrate in the following the principal aspects that influence decisions:

- persistence of faults: faults originating error states may be persistent, temporary, or intermittent;
- context variability: systems with a variable context require an adaptive behavior more than in the case of stable context situations;
- origin of faults: faults may occur at the application or at the infrastructure level;
- service evolution: services may present a be more or less well defined and stable interface to interacting partners;
- service compositions: service can be used in isolation, or in a service composition, which can be more or less dynamic.
- OTHERS???

Concerning quality, adaptation actions can be of two main types: negotiation and repair.

Negotiation actions allow to define or redefine the quality characteristics of interacting services and infrastructure components with a single step or iterative process which allows the definition of new quality thresholds which are acceptable for all participating components in a service composition. Negotiation can be performed at design time, in particular when a set of predefined components participate in a composition, thus preparing the potential components in a composition, as proposed in [24]. In variable context and evolving services a more dynamic approach to negotiation can be envisioned, based on on-line auctions [**?**].

Repair actions are proposed to repair system errors. At the service level, the main repair actions are *substitution* and *retry*, which may need to be executed in a repair plan to maintain a consistent system state, involving for instance also *compensation* actions[25]. Other actions have been proposed for the infrastructural level, mainly based on reconfiguration of services or infrastructural components [**?**]. The reasons for faults at the infrastructure (e.g., Grid) level can be manifold; the geographically widespread nature of infrastructures lead them vulnerable to connectivity problems and the variations in the configuration of different systems, resources that have unpredictable behaviour, problems with the connecting infrastructure or systems just running out of consumable resources (i.e., memory or disk space) are some other possible sources of failure. When discussing the repair actions of an infrastructure, how it recovers from one of these faults is an important factor. Thus, automated procedures may be in place to recover from faults. For example, a task may be retried (possibly on an alternate resource) or the task can be restarted from a checkpoint taken before the task failed.

Decision mechanisms for applying repairs on the basis of the aspects listed above have been proposed, including rule based approaches [**?**,**?**] or automatic plan generation [25]. In general, only the service level is examined in such approaches, while decisions at the infrastructural level are taken independently

to manage the networked system resources. An interaction to manage quality across levels is being proposed in [26] for energy management.

However, the complexity of such decisions might require completely new approaches in future research, since a complete description and control on all system's variables might result impractical. Proposed approaches include automatic classification and learning and the application of systems theory to control the stability of the system.

7 Concluding remarks

References

1. Ran, S.: A model for web services discovery with qos. SIGecom Exch. **4**(1) (2003) 1–10
2. Tondello, G.F., Siqueira, F.: The qos-mo ontology for semantic qos modeling. In: SAC '08: Proceedings of the 2008 ACM symposium on Applied computing, Fortaleza, Ceará, Brazil, ACM (2008) 2336–2340
3. Kritikos, K., Plexousakis, D.: Semantic qos metric matching. In: ECOWS '06: Proceedings of the European Conference on Web Services, Zurich, Switzerland, IEEE Computer Society (2006) 265–274
4. Anbazhagan, M., Nagarajan, A.: Understanding quality of service for web services. IBM Developerworks website (January 2002)
5. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, Budapest, Hungary, ACM (2003) 411–421
6. Avizienis, A., Laprie, J.C., Randell, B.: Fundamental concepts of dependability. Technical Report 0100, Computer Science Department, University of California, Los Angeles, LA, USA (2001)
7. Kopetz, H.: Real-Time Systems: Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers (1997)
8. Cappiello, C.: The Quality Registry. In: Mobile Information Systems – Infrastructure and Design for Adaptivity and Flexibility. Springer-Verlag (2006) 307–317
9. Kritikos, K.: Qos-based web service description and discovery. Phd thesis, Computer Science Department, University of Crete, Heraklion, Crete, Greece (2008)
10. Truong, H.L., Samborski, R., Fahringer, T.: Towards a framework for monitoring and analyzing qos metrics of grid services. In: International Conference on e-Science and Grid Computing, Amsterdam, The Netherlands, IEEE Computer Society Press (December 2006)
11. Lee, K., Jeon, J., Lee, W., Jeong, S.H., Park, S.W.: Qos for web services: Requirements and possible approaches. World Wide Web Consortium (W3C) note (November 2003)
12. Sabata, B., Chatterjee, S., Davis, M., Sydir, J.J., Lawrence, T.F.: Taxomomy of qos specifications. In: WORDS '97: Proceedings of the 3rd Workshop on Object-Oriented Real-Time Dependable Systems - (WORDS '97), Washington, DC, USA, IEEE Computer Society (1997) 100–107
13. Tian, M., Gramm, A., Nabulsi, M., Ritter, H., Schiller, J., Voigt, T.: Qos integration in web services. Gesellschaft fur Informatik DWS 2003, Doktorandenworkshop Technologien und Anwendungen von XML (October 2003)

14. Dey, A.: Architectural support for building context-aware applications. Phd thesis, College of Computing, Georgia Institute of Technology (December 2000)
15. Gray, P.D., Salber, D.: Modelling and using sensed context information in the design of interactive applications. In: EHCI '01: Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction, Toronto, Canada, Springer-Verlag (2001) 317–336
16. Buchholz, T., Küpper, A., Schiffers, M.: Quality of context: What it is and why we need it. In: 10th International Workshop of the HP OpenView University Association (HPOVUA 2003), Geneva, Switzerland. (2003)
17. Sheikh, K., Wegdam, M., van Sinderen, M.J.: Quality-of-context and its use for protecting privacy in context aware systems. Journal of Software **3**(3) (March 2008) 83–93
18. Cardoso, J., Sheth, A.P., Miller, J.A., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. Journal of Web Semantics **1**(3) (2004) 281–308
19. Hielscher, J., Kazhamiakin, R., Metzger, A., Pistore, M.: A framework for proactive self-adaptation of service-based applications based on online testing. In: ServiceWave 2008, to be published (10-13 December 2008)
20. Gallotti, S., Ghezzi, C., Mirandola, R., Tamburrelli, G.: Quality prediction of service compositions through probabilistic model checking. In Becker, S., Plasil, F., Reussner, R., eds.: Quality of Software Architectures. Models and Architectures, 4th International Conference on the Quality of Software-Architectures, QoSA 2008, Karlsruhe, Germany, October 14-17, 2008. Proceedings. Volume 5281 of Lecture Notes in Computer Science., Springer (2008) 119–134
21. Andersen, P.: What is Web 2.0?: ideas, technologies and implications for education. JISC Report (2007)
22. Drago, M.L.: A reputation management framework for web services. Master's thesis, Politecnico di milano (2008)
23. Bianculli, D., Binder, W., Drago, M.L., Ghezzi, C.: Transparent reputation management for composite web services. In: International Conference on Web Service (ICWS). (2008) To appear.
24. Ardagna, D., Comuzzi, M., Mussi, E., Pernici, B., Plebani, P.: PAWS: A framework for executing adaptive web-service processes. IEEE Software **24**(6) (Nov.–Dec. 2007) 39–46
25. WS-Diamond team: DIAMOND Web Services - DIAgnosability, MONitoring and Diagnosis. In: At your service: An overview of results of projects in the field of service engineering of the IST program MIT Press Series on Information Systems, MIT Book). (in press)
26. Ardagna, D., Cappiello, C., Lovera, M., Pernici, B., T.M.: Active energy-aware management of business-process based applications. In: ServiceWave. (2008)