Towards Adaptive Management of QoS-aware Service Compositions - Execution Strategies[1]

Mariusz Momotko[1], Michał Gajewski[1], André Ludwig[2],
Ryszard Kowalczyk[3], Marek Kowalkiewicz[4], Jian Ying Zhang[3]

[1] Rodan Systems S.A., ul. Pulawska 465, 02-844 Warsaw, Poland
{Michal.Gajewski, Mariusz.Momotko}@rodan.pl

[2] University of Leipzig, Marschnerstr. 31, 04109 Leipzig, Germany
ludwig@wifa.uni-leipzig.de

[3] Swinburne University of Technology, PO Box 218 Hawthorn, Victoria 3122, Australia
{jyzhang, rkowalczyk}@it.swin.edu.au

[4] Poznan University of Economics, Al. Niepodleglosci 10, 60-967 Poznan, Poland
M.Kowalkiewicz@kie.ae.poznan.pl

Abstract. Service compositions enable users to realize their complex needs as a single request. Despite intensive research, especially in the area of business processes, web services and grids, an open and valid question is still how to manage service compositions in order to satisfy both functional and non-functional requirements as well as adapt to dynamic changes. In this paper we describe an approach towards adaptive management of QoS-aware service compositions. This approach integrates well known concepts and techniques and proposes various execution strategies based on dynamic selection and negotiation of services, contracting based on service level agreements, service enactment with flexible support for exception handling, monitoring of service level objectives, and profiling of execution data.

1 Introduction

Users want to realize their needs as simply as possible and, therefore, look for sophisticated services that would handle compound needs related to their life events or business activities as a single request. One of the most promising approaches for such sophisticated services is to implement them as compositions of other, simpler services (referred further to as atomic services).

In the last decade, huge effort has been put into developing solutions targeting management of service composition, especially in the area of web services and grids. As its result, a number of new standards on various aspects of service composition management have been defined. In Web Service Business Process Execution Language (WS-BPEL) OASIS standardised a language to define service

[1] This paper presents results of the Adaptive Services Grid (ASG) project (contract no 004617, FP6-2003-IST-2) funded by the Sixth Framework Programme of the European Commission.

compositions. In Web Service Quality Model (WSQM) OASIS also proposed a quality model and a set of quality factors for web services. One of the recent OASIS standards is Web Service Distributed Management (WSDM) which enables management applications to be built using Web services, allowing resources to be controlled (and monitored) by many managers through a single interface. Now, OASIS is working on Web Service Quality Description Language (WS-QDL) which will describe WSQM in a standardised type of XML representation. At the same time, IBM corporation defined Web Service Level Agreement (WSLA) – a language to represent service level agreement (SLA) for web services. Recently, GRAAP working group of the Global Grid Forum has prepared a draft version of a WS-Agreement specification which describes domain-independent elements of a simple contracting process, extensible by domain specific elements.

Focusing on adaptive management of service compositions, several intensive research works have been carried out recently. The *eFlow* project at HP labs [CIJKMMC00] proposes an adaptive and dynamic approach to manage service compositions focusing on their functional aspects such as dynamic service discovery and ad-hoc changes. The *QUEST* framework [GNKCW03] extends the work done on eFlow introducing quality of service (QoS) provisioning. In this framework, contracting of service compositions and atomic services is done by SLA documents. The MAIS project [CP05] focused on negotiation of web service QoS parameters with the ability to use different negotiation strategies. In the area of SLA-based contracting and monitoring, there are several advanced approaches and frameworks such as those presented in [SB04] and [BGO06].

Despite all this efforts, still an open and valid question is how to manage service compositions in order to satisfy both functional and non-functional requirements properly as well as adapt to dynamic changes. So far, adaptability in the existing approaches and tools is weak or inadequate. They do not work well in case of dynamic changes related to the contracted atomic services. In case of failures they have problems with finding alternative solutions that would satisfy both functional and non-functional requirements. In particular, they are not able to: re-negotiate a contract in case of QoS constraint violation, and re-select dynamically another atomic service that satisfies QoS constraints. In addition, the existing approaches do not pay too much attention on service profiling and historical execution data and therefore they are not able to optimise their way of working.

To address this problem we propose a comprehensive approach towards adaptive management of QoS-aware service compositions. It integrates well known concepts and techniques and proposes various **execution strategies** based on dynamic selection and negotiation of services included in a service composition, contracting based on service level agreements, service enactment with flexible support for exception handling, monitoring of service level objectives, and profiling of execution data. Also in the area of technologies we integrate existing solutions such as agent technologies for negotiation, SLA management, business process/workflow management, and QoS monitoring based on SLAs.

The paper is organised as follows. To introduce the approach and show its potential benefits, section 2 presents a usage scenario based on a dynamic supply chain for internet services. This scenario is implemented in the Adaptive Service Grid (ASG) project [ASG]. Then the approach is described in detail in section 3. This section defines execution strategies for service compositions identifying main tasks and showing how these tasks may be organised to assure adaptability. In addition this section describes basic representations of service compositions. The paper closes with a conclusion and an outlook of future work.

The concepts proposed in this article have been used to design an open architecture for adaptive service composition management (presented in another article submitted to ICSOC'2006, Industrial track). This architecture has been implemented prototypically in Adaptive Service Grid, a European co-founded project.

2 Example Scenario

To introduce the approach and to show its potential benefits we start from a simplified description of a Dynamic Supply Chain for Internet services provision (referred later to as DSC scenario), which is a usage scenario that describes a system for automated Domain Name registration and provisioning of Webspace, based on the ASG platform. Services include Domain Name checking, creation of web hosting accounts, registration of domains and so forth. A complete scenario description can be found on the ASG project website [ASG]. Here, we blind out details such as dynamic composition of new supply chain processes and focus on integration of quality of service aspects.

In the DSC scenario composite services composed of a number of atomic services are offered to the end customer. Since individual service requests differ in terms of service quality requirements, execution cost limitations etc. the ASG platform needs to dynamically decide which composition of services is most valuable to a requester. A sample request may be "Provide a service for Domain Name registration, Webspace provision, payment etc. whereas the service needs to be available in less than 60 time units (TU) and costs for setting up this services must be below 15 monetary units (MU)" (initial state request and goal). Based on this information the ASG platform composes a composite service specification that can fulfil the initial state request. Figure 1 shows the composite service specification composed by ASG for the received request.

Services such as *CheckDomain* can be provided by several candidate service providers (e.g. *Denic, UnitedDomains*, and *domainPro)* while other services can only be provided by one service provider (e.g. *CreateWebhostingAccount).* Beside that, some service providers are able to adjust QoS values like duration of execution of their services while others are not. In some cases, duration of services may be dependent on various external factors, and therefore not fixed. By analogy, some of the service providers are able to quote different prices/costs while others only accept fixed prices/costs. Hence, QoS parameters and costs are variable or fixed and

negotiable or non-negotiable. The *CheckDomain* service can be delivered at different QoS levels, i.e. different durations (*DurationX, DurationY*, etc.) and costs (*CostX, CostY*, etc.), and various combinations of them while the *CreateWebhostingAccount* service has a fixed duration of less than 5 TU and costs of 0.50 MU.
Since a service composition specification includes neither conditional nor alternative branches the ASG platform decides to apply the *first-contract-all-then-enact* strategy (compare rules for selection execution strategies given in table 2). Since a composite service specification is just a logical composition of necessary services specifications in a second step the ASG platform needs to select executable service implementations from a potentially high number of service providers before it is able to enact the service composition. In order to fulfil this complex task of end-to-end QoS management and service selection, the ASG platform makes use of negotiation for finding the most suitable configuration of QoS parameter values for all included DSC services.

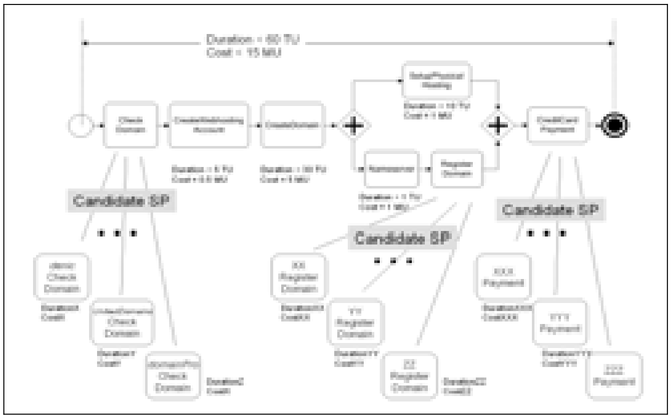

Fig. 1. End-to-end QoS requirements in composite services (before negotiation)

One of the main problems in this area is the decomposition of QoS parameters onto the services involved in the composite services and their optimization to fulfil the reseller's business requirements. In the example in figure 1 the overall maximum duration is less then 60 TU, the *CreateWebhostingAccount* service takes up to 5 TU, the *CreateDomain* service takes up to 30 TU and so forth. Negotiation must find service implementations for *CheckDomain*, *RegisterDomain*, and *CreditcardPayment* that take as little time that the maximum duration threshold of 60 TU holds. Figure 2 presents a possible solution found by negotiation to meet the above described constraints.

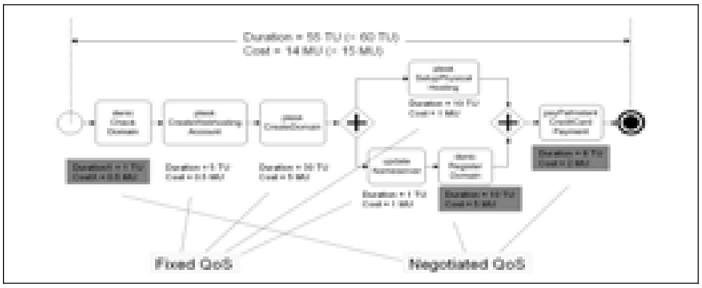

Fig. 2. Duration configuration that meets overall duration threshold (after negotiation)

As stated above the services involved in the DSC scenario are provided externally by service providers. Hence, the possibility to maintain the quality of service provision is outside the influence of the ASG service composition provider and the responsibility of external service providers. In consequence it may happen that services are not satisfying in means of agreed quality, i.e. duration of the *denic:CheckDomain* service is higher than 1 TU. While individual compensation mechanisms such as payment of penalties are an important instrument to deal with such violations with external service providers, the contractual obligations with the end customer demand urgent exception handling such as dynamic replacement of the faulty service. Since end-to-end QoS requirements are still valid for the service composition, dynamic re-selection mechanisms must be applied to find an alternative candidate for the service to be replaced. As an example the *CheckDomain* services can be provided by different service providers as well, i.e. *UnitedDomains* and *domainPro* (see figure 1). *Denic* was initially chosen to be the involved service provider with service duration of 1 TU and costs of 0.5 MU.

The replacement service of the *CheckDomain* service provided by *UnitedDomains* or *domainPro* may not have the same service qualities but it must be ensured that the end-to-end QoS requirements of the service composition are still fulfilled. The process of dynamic re-selection is driven by re-negotiation, essentially a negotiation with the goal of fulfilling overall QoS requirements. Re-negotiation may have chosen *UnitedDomains'* replacement candidate service *CheckDomain* which has a duration of 3 TU and costs of 1 MU which results in 57 TU and 14.5 MU for the overall service composition. The resulting loss in benefits of the service composition provider may at least partially be compensated by penalties claimed from the service provider *Denic* of the faulty service.

3 Execution Strategies for Service Compositions

After introducing the main concepts via example scenario, now we would like to focus on 'dynamics' of service execution and discuss execution strategies. We start from describing the basic tasks which are common to all presented strategies. Then we present the strategies themselves discussing their advantages and disadvantages. Afterwards, we identify several simple rules for the selection of the most appropriate strategy to execute a service composition. Finally, we describe basic representations of service compositions based on WS-BPEL.

3.1 Basic Tasks

There are many possible strategies for executing a service composition. However, there is a common feature of all possible strategies - they are built on the top of basic tasks. These tasks[2] concern a single atomic service and are described in the consecutive sections.

3.1.1 Service Selection and Contracting (SC)

Before execution, a service included in a service composition is expressed as an atomic service specification (see section 3.4), not a concrete service implementation. This specification describes the functional and non-functional requirements of the service as well as the structure of its input and output parameters. In general, there may be more than one atomic service implementations that match this specification. Therefore, we need to have a mechanism to select the most appropriate service implementation. This selection is done as follows. First a service registry is asked to provide a list of all possible service implementations that match functional requirements of a given service. In the next step this list is filtered (through comparison or negotiation) and those implementations which do not satisfy non-functional requirements (i.e. QoS constraints) are rejected. The non-functional requirements for an atomic service are determined on the basis of non-functional requirements defined for the whole service composition as well as information on QoS parameters if already contracted for other atomic services included in the composition. According to the selected strategy, the algorithm for calculation of the QoS constraints for an atomic service may vary. In addition, some QoS constraints may depend on QoS parameters extracted from service profiling data. For example, the QoS constraints defined for a given atomic service may be as follows: *execution cost* less than 5 MU, *duration* not greater than 10 TU and *provider reliability* higher than 85%.

Moreover, some services may declare (during service registration) that some/all QoS parameters which they provide may be **negotiable**. This is especially useful when there are some service implementations that satisfy functional requirements but none of them satisfies all non-functional requirements. In that case the service composition provider can negotiate some/all QoS parameters with the atomic service providers. For instance, we have three possible service implementations: X (0.3 MU $\leq$ execution cost $\leq$ 1 MU, execution duration $\leq$ 1 TU, both negotiable), Y (execution cost = 2 MU, duration $\leq$ 0.5 TU, both not negotiable), Z (execution cost = 3 MU, duration $\leq$ 0.2 TU, both not negotiable), and it is needed to have a service implementation that satisfies the following non-functional requirements: cost $\leq$ 0.5 MU and duration $\leq$ 1 TU, then we may try to negotiate with X for its execution cost. Some possible ways how such negotiation may be done: a) ask X provider to lower the cost – maybe at the

2 The tasks which are not directly related to execution of a service composition have been omitted (e.g. registration or un-registration of an atomic service)

moment the service implementation is able to offer a lower price, b) as a) but we offer that if X provider lowers the price, then the service composition provider will select it more often than the other service implementations (i.e. in the next selections).

For negotiation, we use agent-based negotiation techniques [Br05]. Negotiation of QoS parameters for each service is performed with its candidate service providers according to the FIPA Iterative Contract Net Protocol (ICNP) [FIPA] in coordination with each other. The negotiation agents act on behalf of service consumer to negotiate with different service providers, These agents consist of one coordinator agent, and many negotiator agents.. The coordinator agent is responsible for coordinating concurrent negotiations of all atomic services in the service composition. Each negotiator agent (on behalf of service consumer) is responsible for negotiation for one atomic service in the service composition. These negotiator agents are coordinated by the coordinator agent in order to ensure end-to-end QoS. On the service providers' side, all proxy agents are negotiation engines, each representing one service provider. Normally, for one atomic service, there may be more than one service provider who can offer the same functionality (service), with different QoS. The proxy agent interacts on behalf of one service provider with the same service consumer's negotiator to determine QoS values for that atomic service. The negotiation agents follow ICNP and make negotiation decisions according to their private negotiation strategies in order to best satisfy their users' (service consumer) objectives, preferences and constraints on end-to-end QoS. The negotiation model is based on the principles of multi-attribute utility theory, and involves different negotiation strategies [Ch06] such as concessions and trade-offs, as well as on-line and off-line opponent modelling [BK06] based on the profiling data.

3.1.2 Service Execution and Monitoring (EM)

The service implementation chosen during selection and contracting may be **executed**. All service input parameters are evaluated and, together with a reference to the agreed SLA document sent as a request to the service implementation. After completing a request all results are collected and, if appropriate, passed to the next atomic services within the service composition.

During invocation of the service implementation, it is also **monitored**. Periodically (according, for example, to expected duration of the service invocation determined on the previous service executions), the service provider is asked to complete data on QoS parameters which it is responsible to provide (monitor). This data is then completed with the other QoS parameters which are either provided (monitored) by the service composition provider or determined on the basis of already known values of other QoS parameters. All merged information is entered to the SLA document and verified against QoS constraints. If any of the QoS constraint is violated, a service execution exception is thrown. Also, if the service invocation fails because of other reasons (e.g. lack of required resources, runtime exception, etc.), a service execution exception is thrown as well.

No matter what was the exact result of service invocation (execution failure, success etc.), all events and notifications along with workflow logs from the service execution, are sent to a service **profiler**. This profiler uses this data in order to create up-to-date profiles of service implementations based on the data about service instances at two levels: compound service level and atomic service level. When creating profiles, different types of execution data are used. Service profiler does not analyze service results data, however information from service execution (start time, finish time), service exceptions (critical errors, warnings), and service execution states for interleaved monitoring (for example to monitor execution duration time, to check if SLA agreement has been broken). So far, we consider the following profiling parameters (see table 1). Provider reliability and provider accessibility informs about average reliability and accessibility of all services offered by a particular provider.

Table 1. An initial list of profiling parameters

Parameter name (compliant with WSQM)	***Value type***
ResponseTime (Avg/Min/Max)	Time (ms)
ExecutionDuration(Avg/Min/Max)=ResponseTime+Network delays	Time (ms)
Price (Avg/Min/Max)	Cost (MU)
Reliability = #failed executions / #total executions	Percentage
Accessibility = #successful executions / #total executions	Percentage
ProviderReliability / ProviderAccessibility	Percentage

3.1.3 Exception Handling (Ex)

If a service execution exception is thrown, then the service composition provider tries to find another atomic service implementation which can satisfy functional and non-functional requirements. It should be underlined, that after such a failure, the non-functional requirements will probably be more restrictive than they were when the service invocation had been started (e.g. constraints on duration). To find another service implementation, we propose the following options. 1) if the source of exception was violation of a QoS constraint, then the service composition provider tries to **re-negotiate** the contract with the current atomic service provider (and possibly with other service providers affected by the exception). The (positive) result of such re-negotiation is an updated SLA document. 2) if it was not possible to re-negotiate the contract or the source of the failure was related to a functional problem, then the service composition provider tries to **re-select a replacement from other atomic service providers** that match the service specification and new requirements. Also new service implementations registered after starting invocation of the atomic service may be taken into consideration. 3) if such reselection fails then the service composition provider tries to negotiate the contract in the same way as it is described for service selection and contracting activity. If neither selection nor negotiation is successful, the Service execution subsystem throws exception. In this case, using **re-planning** features, the service composition provider can try to continue service execution with its new definition as it has been described in details in [We05]. Finally, if after the effort above, an alternative solution still could not be found, then the customer is informed that his/her request can not be satisfied.

3.2 Execution Strategies

The basic tasks described in the previous sections are used to define execution strategies. The number of tasks used in a strategy depends on the strategy itself, and on the number of atomic services included in a given service composition. As usual, there is no one optimal strategy. Every strategy focuses on different aspects of service executions and has both advantages and disadvantages. Two basic (quite opposite) execution strategies (see also [ZBLNH03]) and three intermediate execution strategies are described in the consecutive sections.

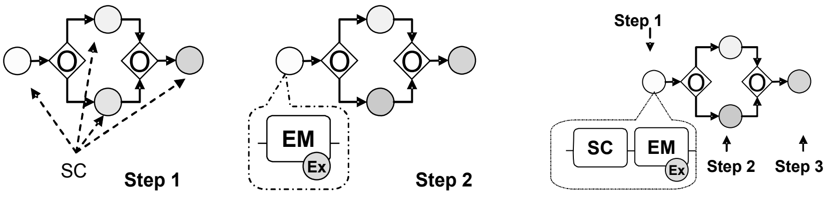

Fig. 3. The concept of the basic strategies: first-contract-all-then-enact (left) and step-by-step-negotiate-and-enact (right).

3.2.1 First-contract-all-then-enact Strategy

This strategy assumes (see figure 3, left side) that selection and contracting of all atomic services included in the service composition is done before its execution. Execution and monitoring of the individual atomic services is done step by step according to the control flow defined for the service composition. Any failure reported during service execution is handled by exception handling mechanism described in the previous sections.

This strategy makes it possible to guarantee non-functional requirements for the whole service composition (global level). Since contracting is done before execution, concurrent selection and negotiation is allowed. As a result, it is possible to consider aggregated concessions and preferences (e.g., if the same provider provides services for several atomic services, then some discount may be regarded), so that the service composition QoS parameters can be optimised.

On the contrary, in this strategy all the activities on conditional branches need to be selected and contracted although some of them may never be enacted. A reservation mechanism is needed. Also service implementations registered during service execution cannot be selected. Finally, the strategy requires coordinated negotiation mechanisms with a coordination agent and a set of negotiation agents.

3.2.2 Step-by-step-contract-and-enact Strategy

This strategy assumes (see figure 3, right side) that selection, contracting and atomic service execution is intertwined. That is, the first atomic service in the service composition can be executed and monitored when its SLA document is established. After completion of this atomic service, the selection and contracting is carried out for each subsequent atomic service and followed by its execution. Any failure occurred during service execution is handled by exception handling mechanism described in the previous sections.

This strategy allows for on-the-fly selection and negotiation based on results and actual QoS values of services that have been executed. This will lead to more accurate and efficient negotiation since it is based on what it had been done for the executed services. Only the invoked atomic services are contracted, not executed branches of service composition are not considered. Also it is possible to select atomic services that have been registered after starting execution of the service composition.
On the contrary, the strategy can only optimise QoS for a given atomic service (local level). As a result, the global QoS requirements can not be optimized. Local constraints need to be provided – but if it gets a list of providers satisfying local constraints then there is a risk of missing their combinations. Instantiation and execution of the whole service composition can not be guaranteed (i.e. a service implementation is executed but it may be impossible to select and contract a subsequent service implementation) thus failing the whole service composition and wasting already executed services (need for un-doing the services).

3.2.3 Other Strategies

The **late-contracting-then-enact strategy** assumes that the selection and contracting of atomic services is done before their execution, as soon as it is sure that they will be executed within a given composition. If, for example, there are two alternative branches, as soon as it is known which of them will be taken, all atomic services on the satisfied branch are selected and contracted. Execution of the atomic services is carried out according to the control flow definition. This strategy is similar to the first-contract-all-then-enact strategy but minimises the risk in contracting services which will never be executed. The risk to not satisfy the global QoS requirements is less than for the mentioned strategy but still exists.

The **first-contract-plausible-then-enact strategy** tries to select and contract first (before service composition execution) all atomic services that belong to the composition path which is the most likely to be executed. The path is predicted on the basis of historical data from previous executions of the service composition. The services that belong to other paths are not selected and contracted. Execution of the atomic services is carried out according to the control flow definition. This strategy minimises the risk of a) contracting services that will never be executed, b) satisfying the global QoS requirements. However, it will work properly only for that cases in which execution concerns the most probable path in the composition. For the other paths it will have similar problems as the step-by-step-contract-and-enact strategy.

The **first-contract-critical-then-enact strategy** selects and contracts before execution only those atomic services which are hard to be contracted dynamically. ‘Hard’ in this context means that the number of service candidates for those service specifications is significantly lower than the number of candidates for the other services included. This strategy is similar to the step-by-step-contract-and-enact strategy but reduces the risk of not satisfying the global QoS requirements. On the other hand, it also does not cope with branches which will never be executed.

3.3 Rules for Selection of Execution Strategies

As was stated earlier, there is no one optimal strategy. However, for some service compositions, there are simple rules which help to select the most appropriate strategy. These rules (see table 2) are based on analysing the control flow of a given service composition, previous executions, non-functional requirements and included atomic services (their profiles).

Table 2. Several simple rules for selection of the execution strategy.

If service composition includes	***Then apply strategy***
neither conditional nor alternative branches	First-contract-all-then-enact
A small # of conditional/alternative branches (e.g. < 10)	Late-contracting-then-enact
just one path with high probability of execution (e.g. 0.9)	First-contract-plausible-then-enact
atomic services which may be easily contracted (i.e. from previous executions we know that for all included atomic services no negotiation was needed).	Step-by-step-contract-and-enact
Significant number of services (e.g. 10) which are 'hard' to contract (please refer to section 3.2.3).	First-contract-critical-then-enact

3.4 Representations for Service Compositions

Specification of a service composition is provided at the design phase of adaptive management of service compositions. This specification describes the composition in terms of control flow (i.e. the order of the invoked atomic services) and data flow (mapping between input and output parameters of atomic services). This specification operates on classes of atomic services, instead of concrete services (compare the DSC specification provided in figure 1). Basically, such a specification provides a solution to satisfy all (static) functional requirements (classes of atomic services that together satisfy them) and leaves flexibility during processing a request (i.e. execution of the composed service) to select and contract concrete atomic services of given service classes that also satisfy its (dynamic) non-functional requirements.

Making the service composition specification generic is very useful, but on the other hand prevents it from using standard execution engines that process WS-BPEL processes. In order to cope with this problem we propose an intermediate solution based on the concept proposed in [AADH]. We represent the specification as a **WS-BPEL process** and, instead of invoking concrete atomic services, we invoke concrete **brokers (services) for atomic service classes**. Every atomic service class has its own broker. Such a broker has input and output parameters as every concrete service of this class. In addition, it receives as an input parameter a set of non-functional requirements that makes it possible to choose an appropriate atomic service of a given class.

The way of using service composition specification at the execution phase depends on the applied execution strategy. For the execution strategies which carry out contracting of more than one atomic service before their enactment (e.g. first-contract-then-enact), the specification will be analysed by service selection and

contracting component (based on WS-BPEL) in order to contract appropriate atomic services. After that, the input parameter representing non-functional requirements will be replaced with information of the selected atomic service and a reference to the agreed contract (SLA document). For the other execution strategies, which intertwine contracting and enactment, the specification will be interpreted by service enactment component (again based on WS-BPEL) which will invoke appropriate service class brokers. These brokers will be responsible for organising selection and contracting and then assure appropriate invocation and monitoring.
History from execution of a service composition is represented as a **service composition execution**. This includes basic information about service composition and invoked atomic services (transformed information from invocation of service class brokers) in the form of a workflow log. This log is used for service profiling.

4 Conclusions

Adaptive management of service compositions is an area of web services research that has recently been attracting more and more attention. The most important questions stimulating research in this area include proper management of service compositions in order to satisfy not only functional requirements, but also non-functional ones. Another important issue is adaptation to dynamic changes in the service environment, which so far has not been researched to a satisfactory extent, and there is still a place for improvement of current methods and tools. The work in the Adaptive Services Grid project, described in this paper, is aiming at solving the stated problems and bringing research results from existing, well developed areas into the field of web services. The work in the ASG project has resulted in implementing the DSC scenario, which shows a potential of adaptive management of services compositions. In our work, in order to propose an open, comprehensive platform for adaptive management of QoS aware service compositions, we integrated concepts developed in different research areas and proposed mechanisms for dynamic selection and negotiation of services, contracting based on service level agreements, service enactment with exception handling, monitoring of service level objectives, and profiling of execution data.

Some issues have still not been resolved. The approaches towards QoS-aware adaptive process management may be lacking some features that are crucial in a real business environment. We are investigating what negotiation features are crucial in a complete platform for web service composition that could be applied in real business scenarios. Another unresolved problem, especially in terms of business applications, is the transparency of the service management platform. In most cases of service use, it is required that reactions are instant and without any delays. The ASG platform, as an entity between atomic service providers and customers has to be as transparent as possible in terms of execution delays, costs and so on.

As future steps and future research directions, we plan to further extend the DSC scenario, include selected Quality of Results parameters that could be potentially negotiated. We believe that proposing a comprehensive solution for adaptive process management, along with proof-of-concept solutions of selected issues will help better understand the nature and challenges of Services Oriented Architecture and in a result facilitate development of SOA applications in the future.

References

[AADH] Aalst, W.M.P. van der, Algred, L., Dumas, M., Hoefstede, A.H.M. ter, Design and implementation of the YAWL system, available via http://www.yawl.fit.qut.edu.au/yawldocs/yawl_system.pdf.

[BGO06] Boström, G., Giambiagi, P., Olsson, T.: Quality of Service Evaluation in Virtual Organizations Using SLAs. submitted to 1st Workshop on Interoperability Solutions to Trust, Security, Policies and QoS for Enhanced Enterprise Systems, 2006.

[Br05] Braun, P., et al.: E-Negotiation Systems and Software Agents Methods, Models, and Applications. In i-DMSS: Foundations, Applications and Challenges. UK, 2005.

[BK06] Brzostowski, J., Kowalczyk, R.: Predicting partner's behaviour in agent negotiation. AAMAS'2006, Japan, May 2006. (in press).

[CIJKMMC00] Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Ming-Chien Shan, M., Ch.: Adaptive and Dynamic Service Composition in eFlow. HP technical Report, 2000.

[CP05] Comuzzi, M., Pernici, B., An Architecture for Flexible Web Service QoS Negotiation, EDOC, 2005.

[GNKCW03] Gu, X., Nahrstedt, K., Chang, R., Ward, C.: QoS-assured service composition in managed service overlay networks. DCS'2003.

[ASG] Integrated Project "Adaptive Services Grid", http://asg-platform.org

[FIPA] FIPA Iterated Contract Net Interaction Protocol Specification available online at www.fipa.org/specs/fipa00030/XC00030F.pdf.

[Ch06] Chhetri, M., et al.: Experimentation with three different approaches of Automation Negotiation. SOCABE'2006, Japan, May, 2006 (in press).

[SB04] Salle, A., Bartolini, C.: Management by Contract, HPL-2003-186, HP labs, 2004.

[We05] Weske, M., et. al.: Dynamic Failure Recovery of Generated Workflows. DEXA'2005, BPMPM Workshop, 2005.

[ZBLNH03] Zeng, L., Benatallah, B., Lei, H., Ngu, A., H., H., Flaxer, D., and Chang, H.: Flexible composition of enterprise web services. Electronic Markets - Web Services, 2003.