Quality Prediction in Service Composition Frameworks

Benjamin Klatt, Franz Brosch, Zoya Durdik, and Christoph Rathfelder

FZI Karlsruhe, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany
{klatt,brosch,durdik,rathfelder}@fzi.de

Abstract. With the introduction of services, software systems have become more flexible as new services can easily be composed from existing ones. Service composition frameworks offer corresponding functionality and hide the complexity of the underlying technologies from their users. However, possibilities for anticipating quality properties of composed services before their actual operation are limited so far. While existing approaches for model-based software quality prediction can be used by service composers for determining realizable Quality of Service (QoS) levels, integration of such techniques into composition frameworks is still missing. As a result, high effort and expert knowledge is required to build the system models required for prediction. In this paper, we present a novel service composition process that includes QoS prediction for composed services as an integral part. Furthermore, we describe how composition frameworks can be extended to support this process. With our approach, systematic consideration of service quality during the composition process is naturally achieved, without the need for detailed knowledge about the underlying prediction models. To evaluate our work and validate its applicability in different domains, we have integrated QoS prediction support according to our process in two composition frameworks – a large-scale SLA management framework and a service mashup platform.

1 Introduction

In recent years, service-oriented systems have gained more and more attention from software providers and consumers. Reduced ramp-up-time, infrastructure cost and management advantages reside on both sides [1]. Meanwhile, service providers do not only offer single services but also composed services on demand, according to individual customer requests. *Service composition frameworks* are emerging [2, 3] allowing for intuitive creation of composed services without requiring expert knowledge about the underlying technologies.

Apart from functional requirements, *Quality of Service* (QoS) properties – such as performance and reliability – are increasingly important to assure user acceptance. Anticipating QoS properties of dynamically composed services is challenging, as it is typically not feasible to create and test each possible composition that a customer might request in a laboratory or field environment. A potential solution to this problem is the use of *model-based software quality*

prediction [4], which does not require the actual execution of services in order to assess them. Rather, it uses simulations or analytical calculations to predict service quality based on the specification of the service architecture and its quality-relevant factors.

However, the missing integration of predictive capabilities in the composition frameworks is a major obstacle to the use of quality prediction approaches. If prediction is applied in isolation, the manual specification of the required input information is laborious and sometimes even impossible due to missing knowledge about the internals of the service architecture. Furthermore, many prediction approaches require to understand the specifics of the employed prediction models. Especially if a customer is interested choosing one of several possible alternatives for service composition, evaluating all of them through repeated manual prediction is highly impracticable.

To overcome this problem, we present a *quality-aware service composition process* in this paper. This process includes QoS prediction for composed services as an integral part and to support the decision between multiple composition alternatives. Furthermore, we describe how composition frameworks can be extended to support our process. The description includes a set of components and interactions that realize the envisioned functionality. We validate our work in two case studies where we integrate QoS prediction support in a large-scale SLA management framework and a service mashup platform. In these case studies, QoS prediction is based on the *Palladio Component Model* (PCM, see [5]).

The contributions of this paper are (i) a quality-aware service composition process, (ii) a schema for integrating corresponding QoS prediction support in service composition frameworks, and (iii) two case studies using a reference implementation based on the PCM.

The remainder of this paper is structured as follows: First, Section 2 provides an introduction to model-based software quality prediction and service composition as relevant foundations for our approach. Our recommended process is introduced in Section 3, followed by the integration in composition frameworks in Section 4. Section 5 describes the provided reference implementation and the two case studies before conclusions are drawn in Section 6.

2 Foundations and Related Work

The work presented in this paper is based on two major research areas. On the one hand, model-based software quality prediction provides a foundation for the predictive capabilities to be integrated in the process. On the other hand, service compositions provide new flexibility for software creation, but also challenges regarding the consideration of QoS properties. In the following, both areas are presented in more detail, followed by a discussion of related work to our approach.

2.1 Services and Service Compositions

Software services are self-describing entities. They encapsulate application functionality and information resources, and make them available through programmatic interfaces [6]. This encapsulation empowers service compositions, which

orchestrate a set of existing services to a new service or application, to provide more complex functionality [7]. The composition of services is described as a workflow using a modelling or programming language, such as the Business Process Execution Language (BPEL) [8] or any kind of glue code. Recent developments aim to ease the service composition to enable people with little or none programming or engineering skills to do this. Service composition frameworks and platforms provide non-programmers with flexible and intuitive general-purpose development environments [9], such as IBM Mashup Center [10], Yahoo! Pipes [11], and DreamFace [12].

The value of composed services is not only assessed in terms of functionality, but also according to their QoS properties, such as response time, throughput or reliability [13, 7]. Service Level Agreements (SLAs) have been introduced to contractually define the terms and conditions of a service delivered to a customer. SLA terms may relate to functional and QoS properties, as well as qualifying conditions (such as expected workload). The major constituent of an SLA is its QoS-related information [6].

2.2 Model-based Software Quality Prediction

Model-based software quality prediction may relate to different QoS properties, such as performance and reliability. Regarding performance, Smith et al. established the term *Software Performance Engineering* (SPE) combining performance analyses with a focus on software design [14]. Meanwhile, the performance engineering community has developed numerous approaches with improved accuracy and usability as surveyed in [15]. Similarly, various approaches exist to predict reliability of software systems and services [16].

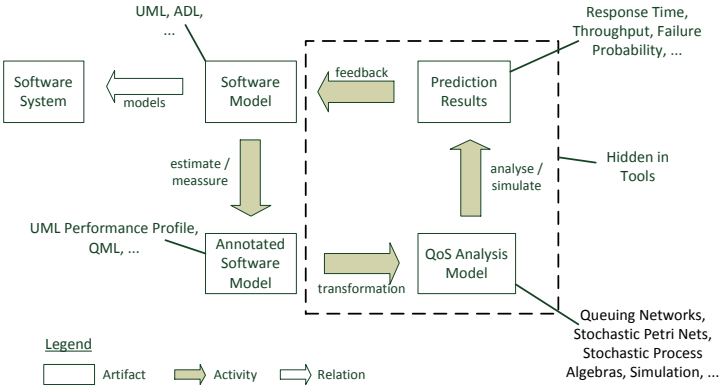

Fig. 1. OMG Quality Prediction Process [17]

A general process for model-based quality prediction has been defined by the Object Management Group (OMG) and is presented in Figure 1 [17]. The starting point of the process is a model of the software itself. In the next step, quality-related annotations are added using either a specific UML profile, such as the UML profile for Schedulability, Performance and Time (UML-SPT) [18], the UML profile for QoS [19], the UML-MARTE profile [20], or a quality-specific

modelling language, such as KLAPER [21] or the Palladio Component Model [5]. The annotated model is then transformed to a QoS analysis model, such as a queueing network [22], Petri net [23], Markov chain [24] or Bayesian network [25], for the prediction itself. The prediction results may be further processed or aggregated if necessary, and returned as a feedback to the modeller, allowing to assess expected QoS levels, and to decide between architectural alternatives.

2.3 Related Work

Various aspects of QoS-aware service composition have been addressed in literature [7], including service discovery and optimization strategies. Klein et al. [26] used linear programming techniques to provide a set of probabilistic service selection policies to chose services at runtime. However, their approach requires runtime information, they are focused on a long-term optimization, and they assume that services can be chosen automatically. Canfora et al. [13] use genetic algorithms to optimize the service selection within a composition. Their algorithm considers QoS as well as cost parameters. However, they are focused on runtime service selection and optimization and do not consider SLA-based quality prediction of service compositions in advance. Garcia et al. [27] describe a framework for service selection based on functional and non-functional properties. However, they are focused on single service discovery and do not consider QoS prediction for service compositions.

A closely related work to our approach is a model-based analysis of service compositions utilizing UML activity models by Gallotti et al. [28]. It describes an assessment of service execution time and reliability properties that can be applied at design time. The activity models which describe the composition are enriched by exploiting a subset of the MARTE UML profile [20]. However, the approach does not consider the automated integration in a service composition framework transparently to the service composer. It also does not process available SLAs to extract quality-related information to merge them into a quality prediction model.

In a preliminary work [29], we describe the details of the QoS prediction integration in the service mashup platform that constitutes our second case study in this paper (see Section 5.3). However, we do not derive a generalized prediction concept and composition process as we do in this paper.

3 Quality-Aware Service Composition

This section introduces our approach of a service composition process with integrated quality prediction. The process describes general steps to create composed services that fulfil given quality requirements. It is designed to consider the generic quality-influencing factors of service compositions, which are presented in Figure 2. The first factor is the service composition workflow (i.e. control and data flow and interactions between involved services) that can have a big impact on the quality of the composed service. The second factor is second, QoS properties of externally required services (e.g., response time and throughput) that can have a significant impact, too. The third factor is the infrastructure used to execute the composed service. Characteristics such as the available memory,

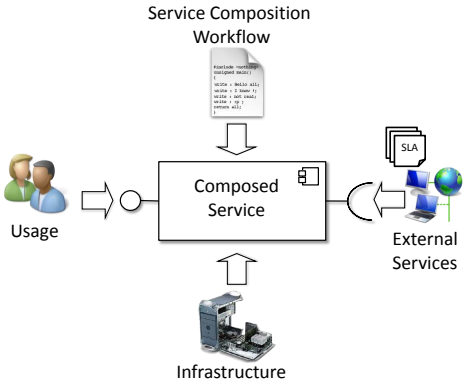

Fig. 2. Service Quality Impact Factors

storage throughput, and cpu performance are relevant for the service execution. Finally, the service usage influences the service quality – for example, a service might perform well for a small user group with a low arrival rate, but it might show a poor performance for a large user group with frequent service requests. Usage characteristics of interest include the expected workload and invocation behaviour, including call frequencies and input data. External service quality, infrastructure quality and service usage profiles can be contractually specified through SLAs and, thus, be automatically taken into account for QoS prediction. Notice that, in order to achieve accurate prediction results, the underlying QoS prediction method should be chosen in a way such that it explicitly takes all described factors into account.

Figure 3 presents an UML activity diagram describing our proposed generic process. This process is conducted by a user of a service composition framework, namely a service composer. The right hand side of the figure shows artefacts related to external services used by the composed service. The left hand side shows QoS requirements that the composed service has to provide. In the central area, the main steps of the process are shown.

The process starts with the specification of the service composition workflow, which can either be created on demand or chosen from a set of predefined workflows. In both cases, the service composition workflow includes dependencies to external software or infrastructure services required to realize and execute the composed service. Based on the specified workflow, the service composer selects external service instances that match the referenced descriptions. There might be multiple instances available per description with different QoS properties, as specified in the corresponding SLAs.

The service selection and the usage profile specification provide SLAs which contain all relevant information to automatically derive a quality prediction model. Based on this prediction model, an automated quality prediction can be performed to provide feedback about the ability of the composition to satisfy the customer's quality requirements. Due to the high envisioned degree of au-

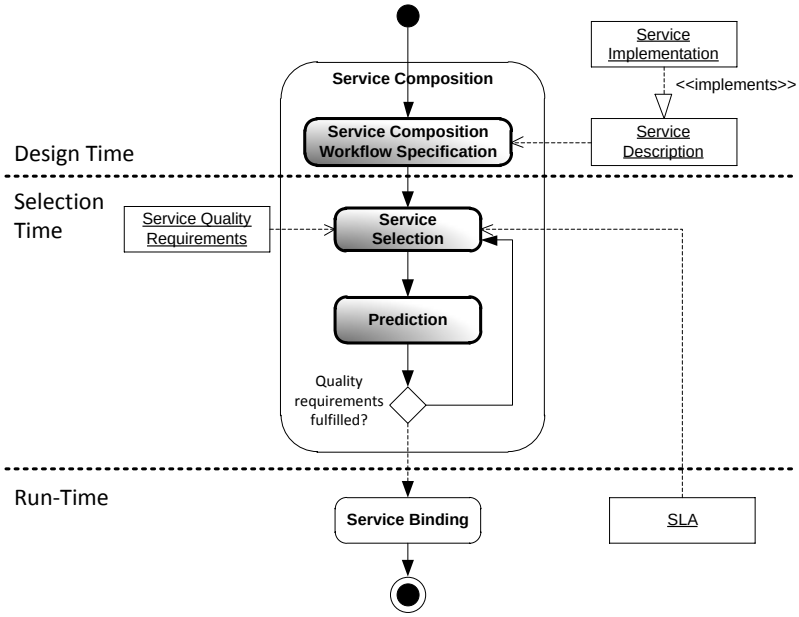

Fig. 3. Service Composition Process: Steps and Artefacts

tomation, the service composer does not need any expert knowledge about the underlying prediction models.

The prediction results are compared with the initial quality requirements. If they are not satisfied, the service composer can select other external service instances with different QoS properties and repeat the prediction. If applicable, the composer can also change the set of considered quality requirements or the workflow specification itself to adapt the preconditions of the scenario. Service selection and prediction may be repeated multiple times, without actually instantiating and executing the composed service, until a satisfying configuration is found. Eventually, the process moves on to the instantiation of the composed service, indicated in Figure 3 through the binding step.

4 Framework Integration

This section discusses how service composition frameworks can be enhanced with predictive capabilities according to our quality-aware service composition process. The prediction functionality is provided by a *Service Quality Prediction* (SQP) component which can be integrated in a composition framework. The details of concrete SQP implementations may vary between different frameworks; however, the conceptional specification as provided here is still valid.

Figure 4 shows how the SQP component is integrated into a composition framework. An important aspect of the integration is the transparency of the QoS prediction details to the user of the framework, namely the service composer. From the composer's point of view, the inputs for the prediction are the specification of a composed service (i.e. the composition workflow specification

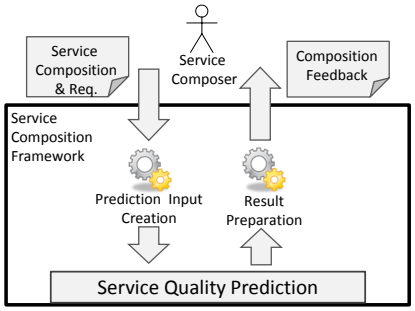

Fig. 4. SQP Integration in Service Composition Frameworks

and the selected external service instances), the envisioned service usage and a set of quality requirements. These inputs are specified in a user-oriented way, through the interface between the framework and the composer. Similarly, prediction results are returned as a feedback to the composer in a form that is most understandable and meaningful to him (e.g., a graphical presentation of predicted QoS metrics or a simple statement if the given QoS requirements are met).

Internally, the composer's inputs are translated to a valid input format for prediction. This translation may include several aspects such as automated interpretation of machine-readable SLA contents and automated creation or adaptation of prediction models. Similarly, prediction results may have to undergo further processing such as aggregation or translation into user-understandable result metrics before they are presented to the service composer.

This decoupling between user- or framework-specific SLA and service description format from one side, and the internal prediction specific representation from another side, supports the usage of proprietary and standard formats (e.g., wsdl, ws agreement, and BPMN). For example, the SLA@SOI framework described in Section 5.2 encapsulates multiple standards. However, even as intermediary format no specific standard could be used as they all have different focus. For example, the ws agreement standard focuses more on the negotiation than the description of quality properties.

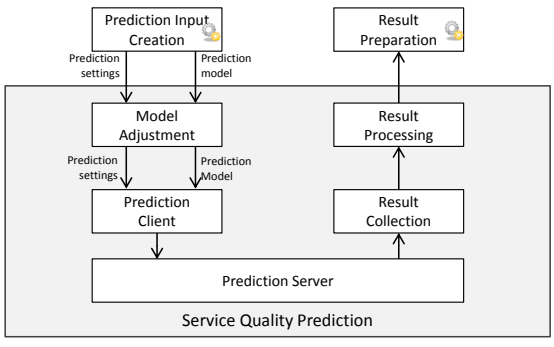

Fig. 5. Internal SQP Structure and Workflow

Figure 5 presents further details of the internal structure and prediction workflow of the SQP component. The prediction input is delivered to SQP in the specific format suited for the applied prediction method by an external *Prediction Input Creation* component. This component acts as an adapter between SQP and the surrounding framework. Typically, the input includes the prediction model itself and further configuration settings, such as required simulation depth or maximum simulation time.

If the available input information is not sufficient to fully create the prediction model, an alternative solution is that SQP loads a predefined base model and adjusts certain parameters of this model according to particular prediction requests. This may be done by an SQP-internal *Model Adjustment* subcomponent. Furthermore, the prediction itself, done by simulation or by analytical solving, may be a time- and resource-consuming task, which should be outsourced to dedicated prediction servers. To this end, SQP may include a *Prediction Client* that sends its requests to a physically remote server. In this case, SQP is a distributed subsystem on its own rather than a single component. Further SQP-internal steps include the collection of prediction results returned by the server and potentially their aggregation (e.g. deriving mean values or percentiles from prediction probability distributions). Finally, the results are returned to the surrounding framework which may further process them for presentation to the service composer.

5 Case Studies

To demonstrate the applicability of our approach, we have integrated QoS prediction according to our described process (see Section 3) in two service composition frameworks. In the following, we first provide an introduction to the prediction method we used for realizing the QoS prediction (Section 5.1). Then, we discuss the two service composition frameworks and how we extended them (Sections 5.2 and 5.3).

5.1 QoS Prediction with the Palladio Component Model

As a prediction method for our case studies, we have chosen the Palladio Component Model (PCM) [5]. While traditionally being tailored towards component-based software architectures, the PCM can also be used to model and predict the performance and reliability of composed services, explicitly taking into account all quality-influencing factors described in Section 3. Its enhanced application of model driven development (MDD) techniques provides a basis for automated transformation of the service composition inputs into prediction models such as queueing networks (QN) for performance simulation or Markov chains for reliability analysis. Realized as stand-alone tooling, the PCM modelling and prediction capabilities have been successfully evaluated in a number of research and industrial case studies [30–33].

To achieve a transparent integration of PCM-based QoS prediction into service composition frameworks, we have extended the existing *PCM Workbench* to offer automated prediction "as a service". We focus on the performance prediction through QN-based simulation. Figure 6 gives an overview of the PCM

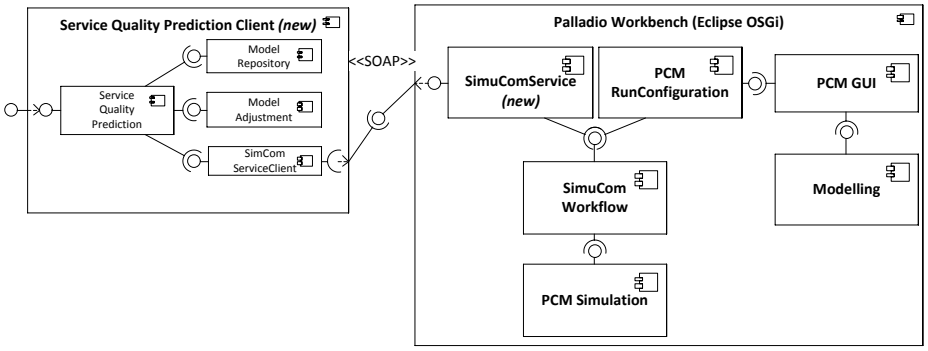

Fig. 6. PCM Extension for QoS Prediction of Service Compositions

components relevant in our context, including existing and newly added ones. For stand-alone QoS prediction, software architects use the *PCM GUI* component to create a design-oriented model of the component-based architecture. The *SimuComWorkflow* transforms the architectural model to a QN and controls the actual prediction carried out through the *PCM Simulation*. The *PCMRunConfiguration* offers various configuration parameters for the simulation. A new *SimuComSerivce* (a SOAP-based web service) has been developed to enable programmatically triggered predictions as an alternative to the manual operation. The service can be remotely invoked and returns aggregated prediction results to its callers.

While the extended PCM Workbench as described here provides the basis for applying the PCM in both case studies, the client side required two individual implementations tailored to the needs of each case study. Figure 6 shows the implementation provided for the first case study (see Section 5.2) as an example. The implementation includes a set of components offering a single SQP interface to the surrounding service composition framework. An internal *Service Quality Prediction* component controls the SQP workflow. A *ModelRepository* stores and loads PCM-based quality prediction models. The *ModelAdjustment* adapts models according to individual prediction requests. The *SimuComServiceClient* encapsulates the remote communication with the *SimuComService* and includes functionality for results collection and processing as shown in Figure 5.

5.2 Case Study I: SLA Management Framework

In the first case study, we integrated QoS prediction in a large scale SLA management framework developed within the SLA@SOI research project [2]. The central goal of the project is to enable a transparent SLA management across the whole business and IT stack. It aims to automate SLA negotiation and to provision, deliver and monitor services in a unified manner with predictable quality attributes that can be enforced at run-time. Based on our approach, service performance and reliability prediction can be applied during the SLA negotiation process to support service providers in creating feasible SLA offers, as illustrated in Figure 7. In addition to the work described in [34], this integration

has been further enhanced and was extended with the experience gathered from the second case study described below.

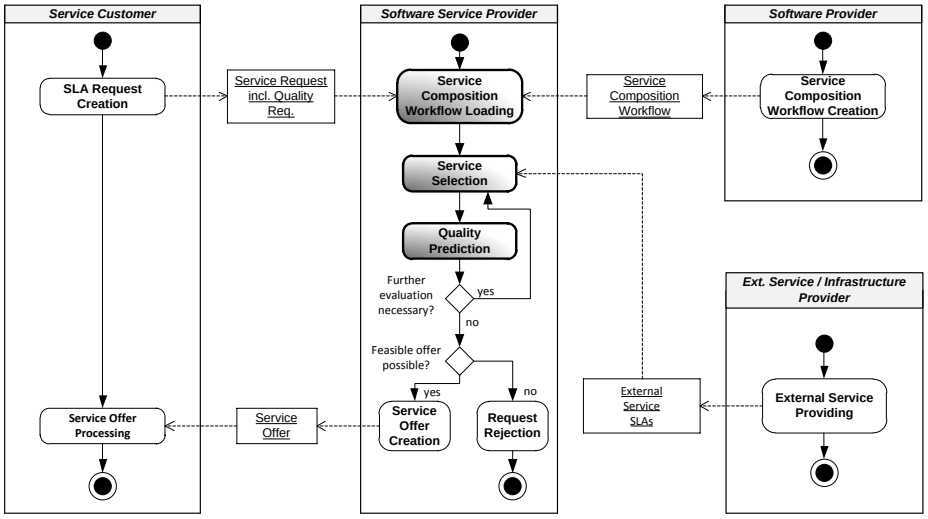

Fig. 7. Quality-Aware Service Composition for SLA@SOI

In the SLA@SOI scenario, a *Software Service Provider* offers a set of software services to *Service Customers* with non-functional property ranges. During SLA negotiation, a service provider and a customer establish a concrete SLA with fixed values for the available property ranges. The provider in turn may act as a customer of other service providers, making SLA negotiation a multi-level process.

The negotiation is initiated by the service customer, who issues an SLA request to the provider. Such a request contains a selection of service operations, as well as a required service quality for a maximum frequency of service invocations. It is the responsibility of the provider to determine if the service can be offered under the requested conditions and for a reasonable price. To this end, he starts our identified general quality prediction process as highlighted in Figure 7. In the SLA@SOI scenario, he first loads the already existing composition workflow provided by the *Software Provider* for the requested service. Then, he selects a certain set of SLAs of the required external services. The latter step includes the selection of appropriate services from external providers, as well as the negotiation of concrete service quality, usage and pricing for these external services. As a third step, the service provider uses the service quality prediction to determine the expected performance and reliability of the target services. Finally, he evaluates the prediction results. If the results satisfy the quality requirements of the SLA request, he returns a concrete SLA offer back to the requesting customer who may accept or decline this offer. If prediction shows that the quality requirements cannot be met, the service provider can

still return a counter-offer containing the predicted quality to the customer, and the customer can still accept this offer. Alternatively, the provider can evaluate other service configurations. If he cannot create any feasible offer in response to the original request, he rejects the request and no service offer is established.

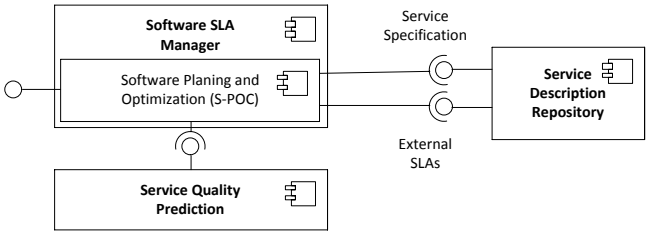

Fig. 8. SLA@SOI Framework Support for Automated SLA Negotiation

Figure 8 shows the parts of the SLA management framework that are involved in the SLA negotiation. QoS prediction is provided by the SQP component as introduced in Section 4. The service description repository comprises the SLA registry and a registry for the available service composition workflows. A *Software Planning and Optimization Component* (S-POC) as part of a *Software SLA Manager* automatically selects service configurations to be predicted on behalf of the software service provider. The SLA negotiation process is not part of the approach presented in this paper; further information about this topic can be found in [35]. The automation removes the burden of manually finding good configurations from the provider, and hides the complexity of the underlying multi-level SLA negotiation process.

In the context of the SLA@SOI project, the capabilities of the SLA management framework have been demonstrated and validated on several industrial use cases covering different application domains, such as ERP hosting, public telecommunication services, Enterprise IT management and eGovernment. Service performance and reliability prediction has been specifically conducted in the ERP hosting and eGovernment use cases, as well as an open source demonstrator centred around a sales services solution. Further details about these use cases have been published in the SLA@SOI deliverables at [2]. The results show that our approach could be applied to successfully integrate quality predictions in the SLA@SOI framework.

5.3 Case Study II: Service Mashup Platform

In the second case study, we integrated QoS prediction support in the service mashup platform envisioned by and realized within the COCKTAIL research project [3]. The platform provides methods and tools for the creation, composition and commercial marketing of light-weight services, which are mostly referred to as *service mashups*. Integrated aspects of the platform include service enabling and operation, logging, user management and accounting based on flexible pricing models. Our extension of the platform with QoS prediction capabilities enhances service composition by informing service composers about the quality properties to be expected for their specified compositions.

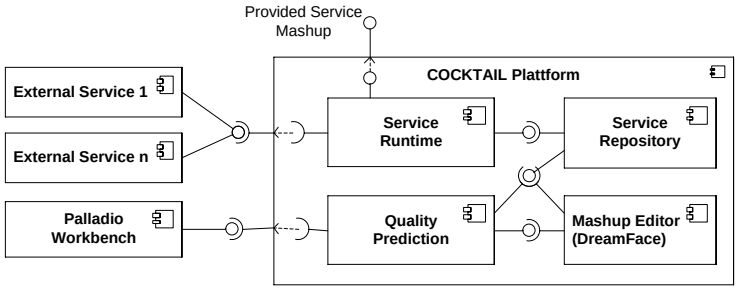

Fig. 9. COCKTAIL Service Platform (excerpt)

Figure 9 shows the parts of the platform involved in the service composition, namely *ServiceRepository*, *MashupEditor*, *ServiceRuntime* and *QualityPrediction*. The *ServiceRepository* contains information about the services that are available on the platform and their properties, such as the type of service, input and output parameters, pricing models and quality attributes. The *MashupEditor* enables the composition of services available in the platform. An existing *MashupEditor* named DreamFace [12] has been adopted for the use in the platform. The *ServiceRuntime* component provides the execution environment for service operation. The *QualityPrediction* component provides the QoS prediction support in the platform. It is invoked by the *MashupEditor* and enables the prediction of the service quality according to the given service composition workflow specification, the specified infrastructure and the service usage profile.

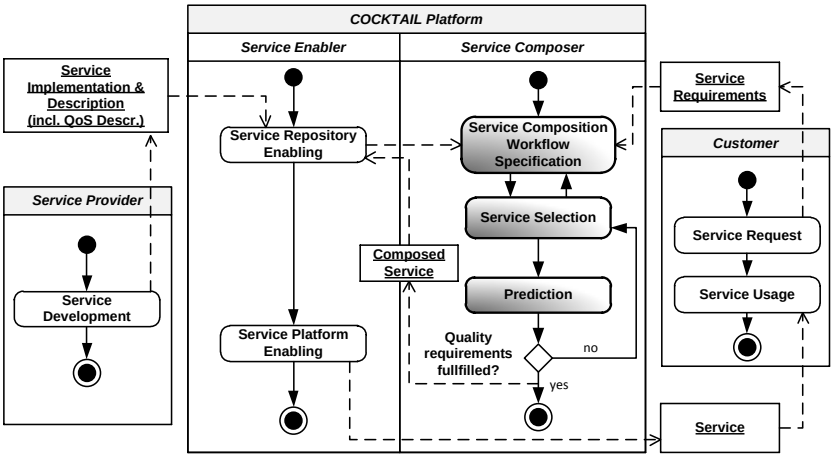

Fig. 10. Quality-Aware Service Composition for COCKTAIL

Figure 10 presents the top-level process of service composition and quality prediction in COCKTAIL. There are four roles involved in the service composition: *Service Provider*, *Service Enabler*, *Service Composer* and *Service Customer*. Service Providers develop services for the platform and provide service

descriptions including quality-relevant characteristics. Service Enablers store the services in the *ServiceRepository* and enable their operation. Potential customers issue service requests which may be associated with specific functional and QoS requirements. If a Service Customer's request cannot be satisfied with an already existing service, the request is sent to a Service Composer to create a corresponding new service. As highlighted in Figure 10, this is done according to the general process proposed in this paper. First, the Service Composer specifies a composition workflow and selects services from the repository. Specific to COCKTAIL, those two steps are done iteratively within the *MashupEditor*. Next, the prediction is conducted to get feedback about the quality properties of the service. This feedback is used to assess the service based on its quality and cost properties. As soon as a satisfying service configuration is found, the composed service is created and stored in the *ServiceRepository* bundled with its specification and quality description. The Service Enabler publishes the service to be operated in the platform and used by the customer.

The COCKTAIL service mashup platform with integrated quality prediction according to our approach has been successfully demonstrated and evaluated on a CRM (Customer Relationship Management) scenario [29].

6 Conclusions and Outlook

In this paper, we have introduced a quality-aware service composition process and its integration in service composition frameworks. Our process is designed to consider the quality-influencing factors of composed services, namely the service composition workflow, the external software and infrastructure service quality and the usage profile. The proposed framework integration intuitively allows for consideration of QoS properties during service composition, without the need for expert knowledge about the underlying prediction models. We demonstrated the applicability of our approach in two case studies, where we integrated QoS prediction support according to our process in two service composition frameworks. The implementation includes performance and reliability prediction based on the Palladio Component Model (PCM).

As shown in the case studies, there is a need for an integrated quality-aware service composition done by service composers who are no experts in quality prediction. Our approach enables the integration of existing QoS prediction approaches in service composition frameworks to support this requirement. Furthermore, we structured the service composition process in a way that ensures all required information to be available when needed, such as SLAs of external services required for the prediction.

Currently, we are integrating our implemented PCM extensions into the existing PCM tool set, which is available as an open source project at [36]. Furthermore, we are planning to support and evaluate additional prediction techniques, and we are working on an enhanced integration of quality predictions in composition frameworks. As part of this we will investigate possibilities of using run-time information as an additional information source for the predictions. These improvements shall further enhance and establish a quality-aware service engineering with predictable and readily negotiable QoS levels.

Acknowledgments This work was supported by the European Commission (grant No. FP7-216556) and by the German Federal Ministry of Education and Research (grant No. 01BS0822). The authors are thankful for this support.

References

1. Kingstone, S.: Understanding total cost of ownership of a hosted vs. premises-based crm solution. Yankee Group Report June 2004 (2004)
2. SLA@SOI: Project website. http://sla-at-soi.eu/ (Jun 2011)
3. COCKTAIL: Project website. http://www.cocktail-projekt.de/ (Jun 2011)
4. Balsamo, S., Marco, A.D., Inverardi, P., Simeoni, M.: Model-based performance prediction in software development: A survey. IEEE Transactions on Software Engineering **30** (2004) 295–310
5. Reussner, R., Becker, S., Burger, E., Happe, J., Hauck, M., Koziolek, A., Koziolek, H., Krogmann, K., Kuperberg, M.: The Palladio Component Model. Technical report, Karlsruhe (2011)
6. Zeng, L., Benatallah, B., H.H. Ngu, A., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. **30** (May 2004) 311–327
7. Strunk, A.: Qos-aware service composition: A survey. European Conference on Web Services (2010)
8. Margolis, Ben ; Sharpe, J.: SOA for the business developer : concepts, BPEL, and SCA. MC Press (2007)
9. Soi, S., Baez, M.: Domain-specific mashups: from all to all you need. In: Proceedings of the 10th international conference on Current trends in web engineering. ICWE'10, Berlin, Heidelberg, Springer-Verlag (2010) 384–395
10. IBM: Mashup center. http://greenhouse.lotus.com/ (Jun 2011)
11. Yahoo!: Yahoo! pipes. http://pipes.yahoo.com/pipes (Jun 2011)
12. DreamFace: Mashup editor. http://dreamface-interactive.com (Jun 2011)
13. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: Proceedings of GECCO '05
14. Smith, C.U.: Performance Engineering of Software Systems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1990)
15. Koziolek, H.: Performance Evaluation of Component-based Software Systems: A Survey. Performance Evaluation **67**(8) (August 2010) 634–658
16. Goseva-Popstojanova, K., Trivedi, K.S.: Architecture-based approach to reliability assessment of software systems. Performance Evaluation **45** (May 2001)
17. Becker, S.: Coupled Model Transformations for QoS Enabled Component-Based Software Design. PhD thesis, University of Oldenburg, Germany (March 2008)
18. Object Management Group (OMG): UML Profile for Schedulability, Performance and Time (January 2005)
19. Object Management Group (OMG): UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms (May 2005)
20. Object Management Group (OMG): UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) (May 2006)
21. Grassi, V., Mirandola, R., Sabetta, A.: From Design to Analysis Models: A Kernel Language for Performance and Reliability Analysis of Component-based Systems. In: WOSP '05, ACM Press (2005)

22. Sharma, V.S., Jalote, P., Trivedi, K.S.: A performance engineering tool for tiered software systems. In: Proceedings of the 30th Annual International Computer Software and Applications Conference - Volume 01, IEEE Computer Society (2006)
23. Kounev, S.: Performance modeling and evaluation of distributed component-based systems using queueing petri nets. IEEE Trans. Softw. Eng. **32** (July 2006) 486–502
24. Reussner, R.H., Schmidt, H.W., Poernomo, I.H.: Reliability prediction for component-based software architectures. Journal of Systems and Software **66**(3) (2003)
25. Roshandel, R., Medvidovic, N., Golubchik, L.: A bayesian model for predicting reliability of software systems at the architectural level. In: QoSA. Volume 4880 of LNCS., Springer (2007) 108–126
26. Klein, A., Ishikawa, F., Honiden, S.: Efficient heuristic approach with improved time complexity for qos-aware service composition. In: The 9th International Conference on Web Services, ICWS 2011. (2011)
27. García, J.M., Ruiz, D., Ruiz-Cortés, A., Martín-Díaz, O., Resinas, M.: An hybrid, qos-aware discovery of semantic web services using constraint programming. In: Proceedings of ICSOC '07. (2007)
28. Gallotti, S., Ghezzi, C., Mirandola, R., Tamburrelli, G.: Quality prediction of service compositions through probabilistic model checking. In: Proceedings of QoSA '08. (2008)
29. Durdik, Z., Drawehn, J., Herbert, M.: Towards automated service quality prediction for development of enterprise mashups. In: 5th International Workshop on Web APIs and Service Mashups @ ECOWS 2011, Lugano, Switzerland (September 2011) to appear.
30. Huber, N., Becker, S., Rathfelder, C., Schweflinghaus, J., Reussner, R.: Performance Modeling in Industry: A Case Study on Storage Virtualization. In: International Conference on Software Engineering (ISCE), Software Engineering in Practice Track, ACM (2010) 1–10
31. Rathfelder, C., Kounev, S., Evans, D.: Capacity Planning for Event-based Systems using Automated Performance Predictions. In: 26th IEEE/ACM International Conference On Automated Software Engineering. (2011) To appear.
32. Brosig, F., Kounev, S., Krogmann, K.: Automated Extraction of Palladio Component Models from Running Enterprise Java Applications. In: Proceedings of the 1st International Workshop on Run-time mOdels for Self-managing Systems and Applications, ACM (2009)
33. Martens, A., Becker, S., Koziolek, H., Reussner, R.: An Empirical Investigation of the Applicability of a Component-Based Performance Prediction Method. In: Proceedings of the 5th European Performance Engineering Workshop. Volume 5261 of LNCS., Springer (2008) 17–31
34. Comuzzi, M., Kotsokalis, C., Rathfelder, C., Theilmann, W., Winkler, U., Zacco, G.: A framework for multi-level sla management. In: Proceedings of the 3rd Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing (NFPSLAM-SOC), Stockholm, Sweden (2009)
35. Kotsokalis, C., Yahyapour, R., Gonzalez, M.A.R.: Sami: The sla management instance. In: Proceedings of ICIW '10. (9 2010)
36. Palladio: Project website. http://www.palladio-simulator.com/ (Jun 2011)