Author(s): Falcarin, Paolo
Article title: Service Composition Quality Evaluation in SPICE Platform
Year of publication: 2009
Citation: Falcarin, P. (2009) 'Service Composition Quality Evaluation in SPICE Platform', In: J. Dong et al. (eds.), High Assurance Services Computing, USA: Springer pp 89-102.
Link to published version: http://dx.doi.org/10.1007/978-0-387-87658-0_5
DOI: 10.1007/978-0-387-87658-0_5

Service Composition Quality Evaluation in SPICE Platform

Paolo Falcarin

Politecnico di Torino, Dipartimento di Automatica e Informatica (DAUIN)

C. Duca degli Abruzzi 24, I-10129, Torino (Italy), Paolo.Falcarin@polito.it

Abstract The goal of the SPICE project is to develop an extendable overlay architecture and framework to support easy and quick creation, and deployment of Telecommunication and Information Services. The SPICE Service Creation Environment (SCE) is used by developers to create both basic services and complex service compositions, which are then deployed in the SPICE Service Execution Environment (SEE), that hide the complexity of the communication environment. Along with its functional interface, each service exposes its own non-functional properties (like Response Time, Cost, Availability, etc…) by means of the SPATEL service description language. These properties are defined in an ontology and this chapter will discuss how the SCE helps developers in evaluating a service composition by calculating the aggregated values of such properties.

1. Introduction

Telecommunication services and network features are often tightly coupled, separate, and vertically integrated. This vertical approach has an extremely weakening effect on service provider's ability to develop more complex services that could span over heterogeneous telecom networks and IT services [1].
The common vision for implementing services is now the realization of an horizontal service platform, based on shared services and network enablers, which can be easily deployed in a distributed SEE (Service Execution Environment) and that can be used as basic blocks in a service composition which may cover different operators domains. Under such assumptions, the composition of communication services, content-based services, Internet-like services, and messaging services, which may span over different service providers, can affect the quality of service perceived by users. In fact, system administrators working in an operator domain can apply quality enhancements on services running in their own SEE but they cannot access to a third-party SEE hosting services involved in a service composition.

Innovative model engineering techniques, like Model Driven Architecture (MDA) approach [2], tend to be used to abstract commonality between different execution platforms and to facilitate the development of systems that can target different execution environments. The exploitation of these techniques in the context of service engineering and, more specifically, in the telecommunication domain [3] is perceived as an opportunity for exporting on service interface non-functional properties.
A Service Creation Environment (SCE) is then needed to facilitate the composition of existing services and the semi-automatic configuration and deployment of IT-Telecom services [4]. The benefits of service composition stem from the possibility of reusing the effort invested in developing services, thereby enabling faster time-to-market and lower costs in the service development process. Under such assumptions, the SPICE project has designed and implemented an example of SOA in the telecommunication domain [5] in order to fulfill these requirements.

One of the goal of the SPICE platform is to provide high assurance composed services, even if they are made of services running on different application servers, in different domains.
In the following section the architecture of SPICE SEE is described, followed by a description of the SPICE SCE which helps service developer in evaluating the quality of an orchestration of telecom-IT services taking into account non functional properties; the usage of such properties is then discussed in an overview section about SPATEL language [6]; finally the aggregation of both static and dynamic non-functional properties is discussed with an example, before drawing conclusions.

2. SPICE Project

One of the goals of the SPICE Service Creation Environment is to facilitate the composition of existing services, to build new services. The benefits of service composition stem from the possibility of reusing the effort invested in developing services, thereby enabling faster time-to-market and lower costs in the service development process. This leads to direct and indirect benefits to service developers, platform operators and service providers.
The SPICE SCE provides facilities for designers to perform service composition, with a higher degree of automation than is provided in a traditional graphical service designer tool.
SPICE project has developed a SCE and a SEE to respectively compose and execute both IT and telecom services. The SCE allows developers to build their own service and to annotate its SPATEL representation with non-functional properties; moreover SCE allows developers to compose such services in a workflow of SPATEL services, and to get an estimation of the aggregated values of non-functional properties depending on the service composition workflow.

The SPATEL service description is published in a service repository and its functional part is translated to WSDL [19]. In case of SPATEL service compositions a BPEL script is automatically generated by the SCE and then deployed in the Service Execution Environment for orchestrating different web services running on multiple execution platforms. An overview of the main components of the SPICE architecture is sketched in figure 1.

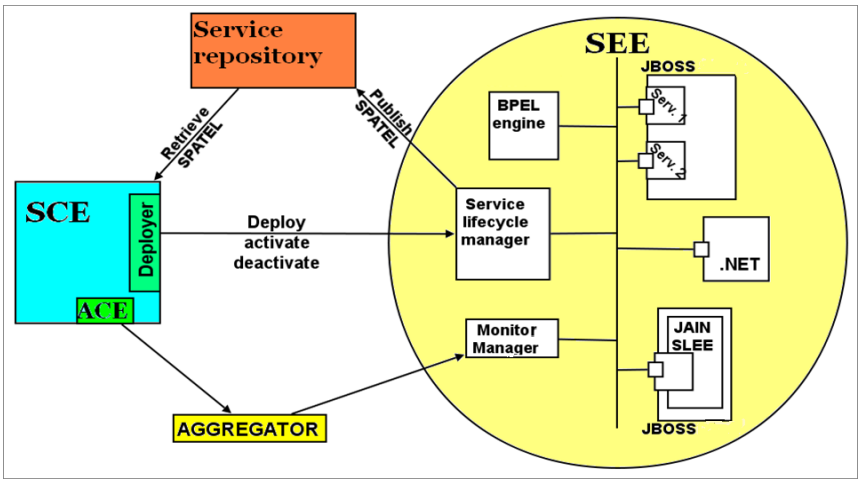

Fig. 1. Main Elements of SPICE Architecture.

The Service Creation Environment (SCE) is used by professional developers for designing arbitrarily complex services by using the SPATEL formalism for high-level design, in combination with general purpose languages for completing the non generated parts of the code of the service. In particular the tool will be used to specify composite services orchestrating other services, which could pre-exist or be developed from scratch.

The SCE provides different pluggable transformers that supports the translation of the SPATEL specification in the interface code for a target execution platform (such as JAIN-SLEE [8], J2EE [9], BPEL [16]). Within the SCE, two components are particularly relevant: the Automatic Service Composition Engine (ACE) and the Deployer.

Deployer is used to package and deploy a SPICE component and/or a SPICE service composition in the target SEE, sending packaged code, WSDL and SPATEL descriptions to the Service Lifecycle Manager which performs the actual service deployment on one of the selected platforms and publish the SPATEL service description on the public service repository.

Service Repository is queried by the SCE to fetch SPATEL descriptions of available services, and these can be used by developer to build a service composition.

For example, the developer specifies in the SCE a service request in terms of its inputs, outputs, preconditions, effects, and some non-functional properties.
This service request is passed to the ACE component which calculates automatically service compositions based on such request: the ACE analyzes SPATEL descriptions of services published in the repository and provides different possible service compositions which may satisfy the desired goal.
Then the developer can evaluate such compositions to see if they actually match the service request (goal), and select one which can be deployed in the SEE as a BPEL script.
In case different compositions match the goal the developer may want to select the one offering the best quality of service: thus SCE can query the Aggregator service to calculate the aggregated values of non-functional properties like response time. This process is sketched in figure 2.

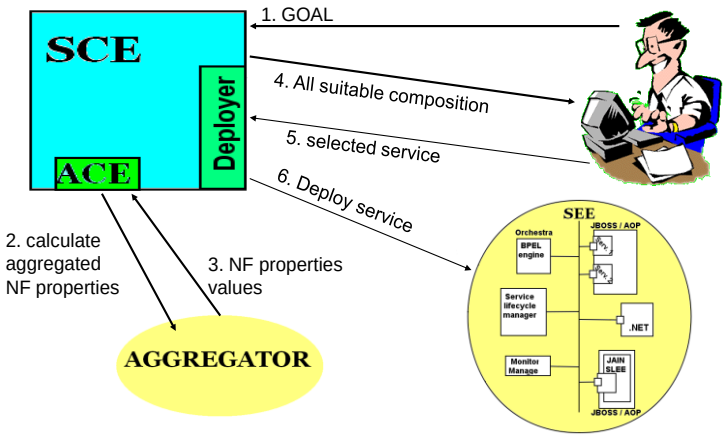

Fig. 2. Service Composition Process.

There has been a lot of interest in defining and working out mechanisms and frameworks for service composition in the industry and the academia [11]. Many approaches use semantic web services, i.e. web services interfaces annotated with semantic tags like WSDL-S [26], or other service description languages [14] [22] are used to more precisely describe information like: provider details, service goal, service parameters' types, service's quality attributes. Annotations follow formal terminologies, which are defined in an ontology, and they are machine-understandable and then usable for being processed by specific tools.
The services can be assembled together by using an automated process based on semantic tags. From this assembly, a "business process", expressing the logic of the calls to "elementary services", is generated. The new composite service is gen-

erated from the business process and later deployed. Goal-based approaches intend to provide a composite service from a request expressed in certain format. The request reflects the goals to be reached by the composite service.
For example, Fuji et al. [13] use "semantic graphs" derived from natural language descriptions, while in [28] semantic interfaces are annotated with service goals are used to compose services. A web services composition methodology is achieved by stitching together semantically-annotated web service components in a BPEL flow [27], while a composition of services as a directed graph where nodes refer to web services was presented in [10].
Aggregation of non-functional service properties then becomes a key decision factor for discriminating among a set of suitable service compositions.
In SPICE SCE service compositions are evaluated for their aggregated non-functional properties, selected and ranked; in this case, generation of alternative valid compositions is a process different from aggregation. Moreover, SCE focuses on checking if a service composition matches certain non-functional properties specified in a required service specification and it allows ranking of viable service compositions based on such properties: examples of non-functional properties that can be considered are cost, response time and reliability.
Yu et al. [30] proposed an approach that selects service components that, composed together, have a desired QoS: as input the request is parameterized with a process (e.g. a BPEL process) that identifies service component types rather than running service components.
This approach is also limited to processes that call service components in sequence, without considering choice nodes and loops in a process. Optimal selection of service components is a key issue for creating a service composition [
31], but service description must provide more information that can be used to drive the developer during the composition design.
Jaeger et al. [15] proposed a mechanism to determine the overall Quality-of-Service (QoS) of a composition by aggregating the quality attributes of the individual services: they identify abstract composition patterns, which represent basic structural elements of a composition like a sequence, a loop, or a parallel execution and they define aggregation functions for each quality attribute: this theoretical approach is used and partially implemented in this work to calculate aggregate values of some non-functional properties, as described in section 5.
For more details on automatic service composition issues see [13] [23].
In following sections we will see how SPATEL language allows defining and exposing non-functional properties.

3. SPATEL Language

The SPICE project has defined a high-level and executable language for describing composite telecommunication services. This formalism, named SPATEL [6], meaning SPICE Advanced language for Telecommunication services, can essen-

tially, be seen as a customization of the UML language for expressing the definition of service interfaces and service composition logic more suitable for the telecom domain. In contrast with most IT web services, telecom services are generally transactional, asynchronous, stateful and sometimes long-running processes, thus it is important to define constraints on the service interface such as the ordering of operation invocations.

SPATEL can be used both to define a single semantic-annotated service interface and to describe an orchestration of components through state machines which are more suitable for integrating "voice-based" dialogs in a service specification, since state machines are the most used paradigm for expressing the complexity that can be found in human-machine voice conversations.

We should note however that the scope of SPATEL is much broader than the scope of traditional voice services since we have to deal with remote synchronous and asynchronous invocations, parallel threads of execution and it provides means to represent typical voice-based interactions, inheriting from previous research work in the field of voice service modeling [3].

The SPATEL formalism aggregates well-know constructs coming from different sources (VoiceXML [http://www.omg.org/spec/QVT/1.0/

18], ITU-SDL [19], SA-WSDL [21]) in order to provide the needed subset that is needed for a high-level and executable formalism usable in telecom context.

SPATEL formalism has been defined by an EMOF metamodel [25] and it comes with a UML2 profile defining the conventions for using the UML graphical notation, used to build service orchestrations.

The service interface description typically publishes the signature of each operation (its input parameters, result and message types), like in WSDL [19] the well-known standard for web services.

Service developer with SPICE SCE can use such additional information to compose a new service made up of an orchestration of different services, typically running in different service providers' domains.

In SPICE platform each service is described by means of the SPATEL language which allows enriching service interface with semantic annotations and non-functional properties which represents instances of concepts defined in a common ontology defined in the SPICE project [7].

Indeed, without a shared understanding (both of semantics and syntax) between applications, the communication is not feasible, or it has to be obtained with manual integration. In this case an ontology is useful, as it is a formal specification of concepts, axioms and definitions stated in a description logic, that enable computers to understand process its content.

For example, the Ontology Web Language (OWL) [24] describes in XML the concepts and their relationships, with different levels of formality, in a particular domain. By establishing a common vocabulary among services, the ontology files support the sharing and reuse of formally represented knowledge.

An important feature of SPATEL is the ability to annotate the elements of the interface (like the operations and the parameters) with semantics tags and non functional properties to enable better service discovery and automated composition.

Non-functional properties are partitioned on the basis of categories like quality of service (QoS), charging, internationalization, etc… The annotation mechanism is similar to the SAWSDL approach [21], as it relies on pointers to concepts defined in external ontologies.

SPATEL language follows an approach to semantically annotate the different aspects of the service, like types, operations and service goal. The following list contains the different kinds of semantic annotations that are present in SPATEL:

- Annotations on IO parameters of the service.
- Annotations on goals that describe the overall objective of a service or the objective of a single operation exposed by the service.
- Annotations on the effects of a given operation that describe the outcomes of its execution in terms of state achieved by the service or action performed.
- Annotations on the preconditions of a given operation describe the conditions that have to be satisfied in order to allow its execution.
- Annotations on non-functional properties to describe aspects related to the quality of service, charging or resource usage.

In figure 3, an example of SPATEL notation for a composed service is shown, and more details on the graphical notation are described in [6].

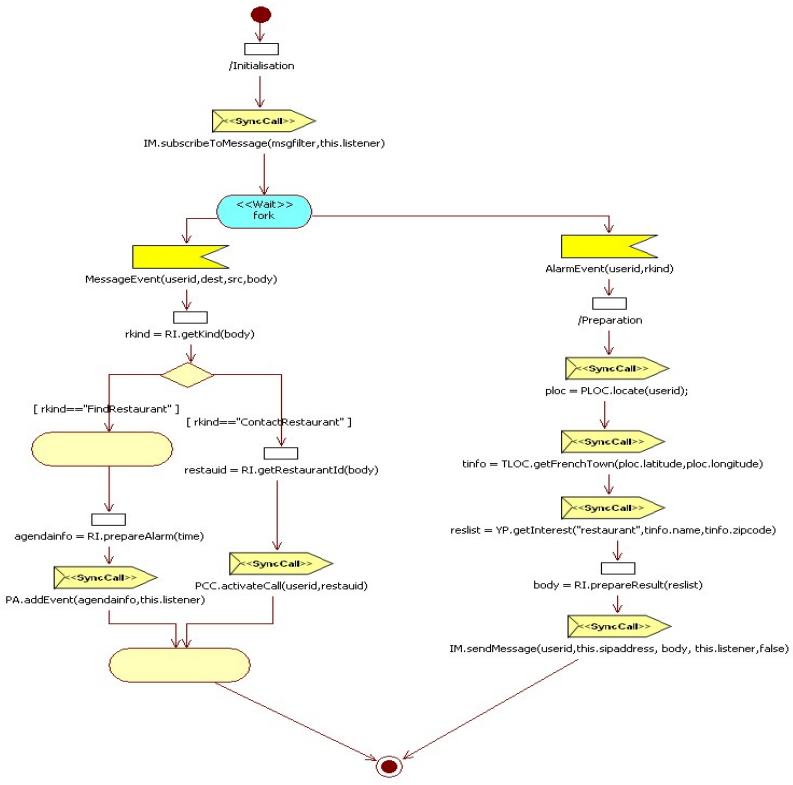

Fig. 3. An example of service composition with SPATEL graphical notation

4. Ontology for non-functional properties

In commercial communication services a very important aspect is non functional properties or quality attributes of services. They are important to provide and guarantee good usability and a good user experience for the service consumer, and they are important for monitoring and control purposes for the service provider. In SPICE project different categories and attributes have been considered and structured into an ontology of non-functional properties (see figure 4).
In contrast to functional attributes, the number of non-functional attributes can be virtually unlimited, and many research works have already been performed to classify them [12]. A similar initiative to categorize non-functional properties is the Web Services Modeling Ontology (WSMO) that attempts to provide a framework to be used for describing web services and their non-functional properties [29]. The framework offers an outline of the type of non-functional properties that are required: error rate, network quality of service, reliability, robustness, scalability, security, transactions and trust. In SPICE many features and concepts of telecom domain has been described which may partially overlap the WSMO: more details on SPICE ontology are available in [7].
The non-functional categories below are the one considered in SPICE, among all those defined in the ontology:

1. Charging: A function whereby information related to chargeable events is formatted, stored, and transferred, correlated, rated and charging accounts are adjusted accordingly. This is necessary in order to make it possible to determine usage for which the charged party may be billed.
2. QoS properties considered are:
   - Response Time: the time a service operation takes to provide a result whenever it is invoked; average, minimum, and maximum response times are considered.
   - Availability: the percentage of time on which a service is operable and ready to provide its capabilities:
3. Security: Encryption types. Security aspects are usually handled separately in a platform. However, some services provide sensitive information that one does not want to pass unencrypted through the network or to other services. From the service composer viewpoint it is important to evaluate which one of the services in a service orchestration does not encrypt data sent on the network, in order to possibly replace it with another similar service offering such security features.
4. Internationalization: in case of information services (like Yellow Pages) it is important to know in which languages the results can be expressed.

The non-functional properties can be divided into two main groups: static and dynamic properties. For example, charging rates, language support are relatively fixed attributes. Even though the total cost will vary, you know exactly in advanced how it will vary. The values are defined manually, so what you read is

what you get. Other attributes like response time will only be an average time provided by the Monitor Manager of the SEE in which the service is deployed.

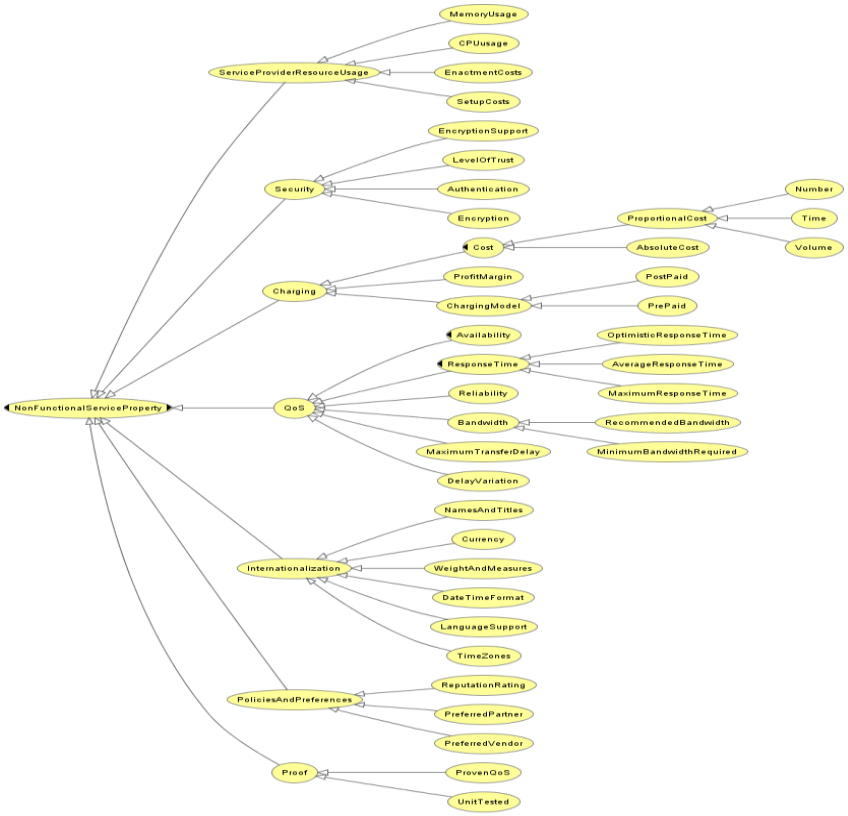

Fig. 4. High-Level View of Ontology for Non-Functional Service Properties

5. Aggregation of Non Functional Properties

The Aggregator defines a pluggable architecture for aggregation of non-functional service properties. Each attribute or attribute category can be addressed in different specific aggregator services which are added to the framework. This allows extensibility of the framework with additional non-functional properties, and enables the reuse of functionality intended to analyze the SPATEL specification which represents the service composition.

The Aggregator assumes that services in a composition do not depend on each other. This assumption states that the result or the execution of one service does not change the quality of other services. Moreover non-functional properties refer to the same definition in the above-mentioned Non-Functional Properties ontology, where their unit of measure is defined along with transformations among different units of measure.
Given a SPATEL representation of a service composition, the Aggregator identifies abstract composition patterns (as defined in the previous work of Jaeger et al. [15]), which represent basic structural elements of a composition, like a sequence, a loop, or a parallel execution.
The aggregation of non-functional properties is based on an algorithm which recognizes composition patterns occurring in workflow and orchestration languages (like BPEL), it applies existing aggregation functions [15], and invokes the specific aggregator components to obtain aggregated values for these patterns.
In case of a decision point where the control flow splits in different separate branches or in the parallel fork case, it is assumed that all branches have the same probability.

For example, three kind of aggregator components have been implemented for the following non-functional properties:

- **Execution Time**: in a sequence, the time is determined by the sum of the values of each involved service. The definitions for minimum and maximum execution times are in a sequential case the same. In case of parallel execution of services the minimum value for execution time is the largest value of all involved services.
- **Cost**: the cost of a service is a measure for the resources consumed by a service execution. Different from the execution time, all services that were used must be taken into account, regardless whether they are relevant for the synchronizing join or not.
- **Encryption Level**: in this case it is assumed that the encryption level is equivalent with the kind of algorithm and related key's length used for signing or encryption, enumerated in a series of discrete values. For the aggregation of the encryption level in a sequential pattern, only the weakest key is significant.

In the example of figure 5, there is a graph representing a service composition obtained from a SPATEL diagram depicted in figure 3, where each node represents a service and an edge between two nodes represents a temporal sequence in the control flow of the service orchestration. In particular, this figure shows the values of non-functional properties.

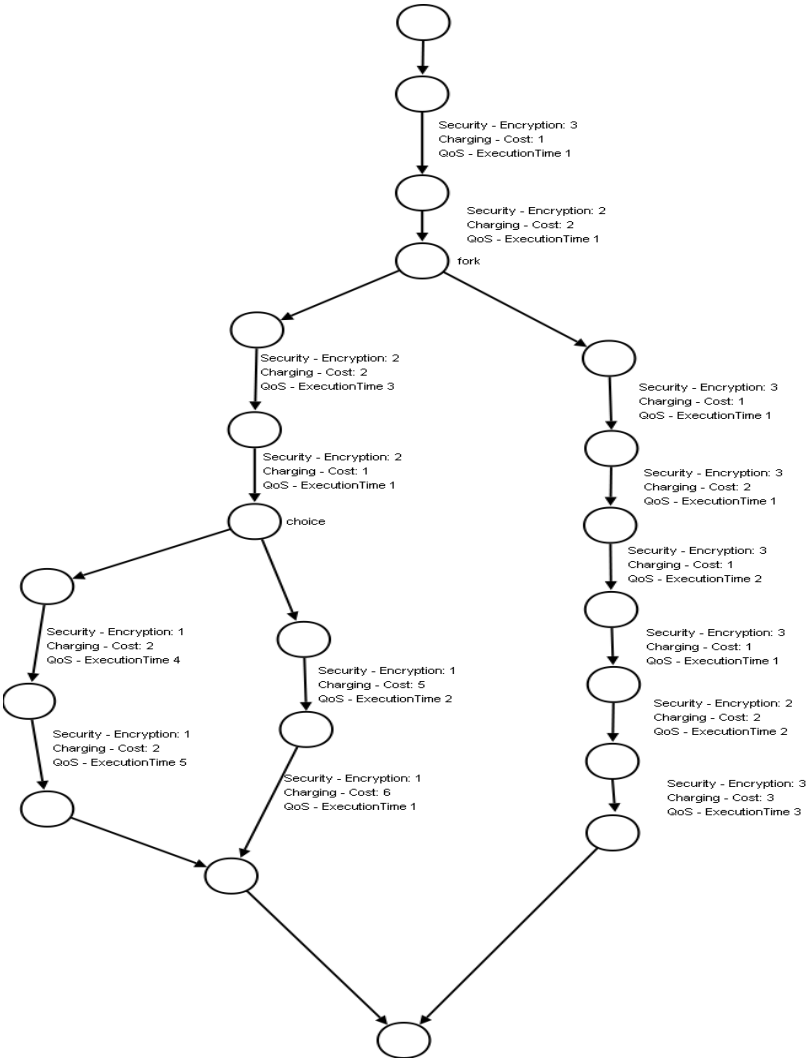

Fig. 5. Initial graph of the service orchestration

The Aggregator can extract this graph from the SPATEL composition and it collapses the nodes in a each sequential path, calculating the aggregated values of the non-functional properties, using the above-mentioned aggregation functions [15]. After a first transformation where all sequential paths are collapsed it is time to collapse parallel nodes in a single one, then the graph is transformed in the one in figure 6; then the algorithm restarts collapsing sequential paths followed by parallel ones. Applying continuously the collapsing of nodes, the graph is reduced to a single node exposing its aggregated non-functional properties.

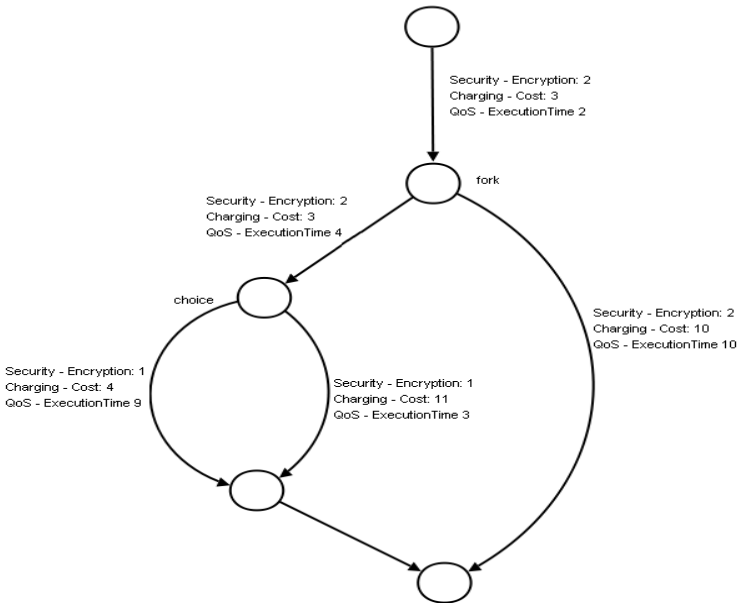

Fig. 6. Service orchestration graph after first aggregation

One of the goal of the SPICE platform is to provide high assurance composed services, even if they are made of services running on different application servers, in different domains, but sharing a common management API to be used by system administrators for monitoring purposes.
The Monitor Manager invokes the management API of each application server in the SPICE SEE, in order to get statistics on response time of each operation of each service deployed in the application server, and it stores these values in its local database. Whenever a system administrator or the Aggregator invoke the Monitor Manager interface, it calculates and returns the requested performance indicators of each service, like values minimum, maximum and average response time.
These information can be used by the aggregator which takes information about the non-functional properties of various services and use these values to return an aggregate of non-functional properties' values.
In SPATEL each non-functional property can be set as static or dynamic.
If the non-functional property is static, then its value has been set by the service provider before deployment and cannot always be trusted, while if the property is dynamic, it means that its value is calculated at run-time querying the appropriate service in the SEE.
The developer can thus choose a determinate service composition, depending either on static properties (like cost or security level) or on dynamic ones calculated on actual values observed by the Monitor Manager in the SEE.

6. Conclusions

SPICE project has developed a SCE and a SEE to respectively compose and execute both IT and telecom services. The SCE allows developers to build their own service and to annotate its SPATEL representation with non-functional properties; moreover SCE allows developers to compose such services in a workflow of SPATEL services, and to get an estimation of the aggregated values of non-functional properties depending on the service composition workflow.
This chapter described how the SPICE project manages the non-functional service properties at design-time, and how the Aggregator service calculates the overall aggregated non-functional properties of a service composition designed by the SCE developer, relying also on the Monitor manager which provides live values of dynamic non-functional properties such as Response Time.
This kind of evaluation of service composition quality attributes is useful for service developer to carefully select services to be bound in a service composition, which will be deployed and executed as a BPEL orchestration script in the SEE.
Future work is devoted to measure the performance and scalability of this approach on large service repositories and more complex service composition workflow structures.

Acknowledgments

This work has been performed in the framework of the IST project IST-2005-027617 SPICE, which was partly funded by the European Union. Special thanks to all project partners, and in particular to Mariano Belaunde (Orange Labs), Alessio Bosca (Politecnico di Torino), Mazen Malek Shiaa (NTNU Trondheim), and Anne Marte Hjemas (Telenor).

References

1 S. Tarkoma, B. Bharat, E. Kovacs, H. van Kranenburg, E. Postmann, R. Seidl, A. Zhdanova, “SPICE: A Service Platform for Future Mobile IMS Services,” in Proceedings of IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM 2007), June 2007, pp: 1-8, ISBN: 978-1-4244-0993-8.
2 OMG, "Model Driven Architecture". Web link: http://www.omg.org/mda/
3 M. Belaunde, J.M. Presso, Vision for an industrial application of MDD in the Telecommunications Industry, ECMDA'05 Conference, Springer July 2005.
4 P. Falcarin, C. Venezia: “Communication Web Services and JAIN-SLEE Integration Challenges”. In Journal of Web Services Research (JWSR), Vol. 5(4), IGI-Global, 2008, ISSN 1545-7362.
5 SPICE (Service Platform for Innovative Communication Environment) project homepage. On-line at http://www.ist-spice.org/
6 M. Belaunde, P. Falcarin, “Realizing an MDA and SOA marriage for the development of Mobile Services”, European Conference On Model Driven Architecture, June 2008, Springer.
7 C. Villalonga, M. Strohbach, N. Snoeck, M. Sutterer, M. Belaunde, E. Kovacs, A.V. Zhdanova, L.W. Goix, O. Droegehorn, “Mobile Ontology: Towards a Standardized Semantic Model

for the Mobile Domain," in Telecom Service Oriented Architectures Workshop (TSOA-2007), September 2007, to appear on Springer LNCS.
8 JAIN-SLEE API Specification. Java Community Process website: http://jcp.org/aboutJava/communityprocess/final/jsr022/index.html
9 Java 2 Enterprise Edition. http://java.sun.com/javaee/
10 Zhang, R., Arpinar, I.B., Aleman-Meza, B.: Automatic composition of semantic web services. In IEEE ICWS. (2003), pp 38–41.
11 M.P. Papazoglou, D. Georgakopoulos, "Service-Oriented Computing," Communications of the ACM, October 2003, Vol. 46, n. 10, pp. 25-28.
12 J. O'Sullivan, D. Edmond, A.H.M. ter Hofstede, "What's in a service? Towards accurate description of non-functional service properties", Distributed and Parallel Databases, 2002, pp. 117-133, Kluwer ed.
13 K. Fujii, T. Suda, "Semantics-Based Dynamic Service Composition," IEEE Journal on Selected Areas of Communications, v. 23(12), December 2005, pp. 2361-2372.
14 Web Ontology Language specification. On-line at http://www.w3.org/2004/OWL/
15 M.C. Jaeger, G. Rojec-Goldmann, G. Muhl, "QoS Aggregation in Web Service Compositions", IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05), 2005, pp. 181-185.
16 BPEL, Business Process Execution Language for Web Services. On-line at http://www.ibm.com/developerworks/library/specification/ws-bpel/.
17 OMG, "MOF 2.0 Query/Views and Transformations", http://www.omg.org/spec/QVT/1.0/
18 W3C/VoiceXML Forum: Voice Extensible Markup Language, www.voicexml.org
19 W3C, Web Service Definition Language (WSDL), www.w3.org/TR/wsdl
20 ITU-T, Specification Definition Language (SDL). Web link: www.itu.int/ITU-T
21 W3C: Semantic Annotations for WSDL and XML Schema, W3C Recommendation, 28 August 2007. Web link www.w3.org/2002/ws/sawsdl/
22 OMG, Uml Profile And Metamodel for Services RFP, http://www.omg.org/cgibin/doc?soa/06-09-09, September 2009
23 A. Bosca, G. Valetto, R. Maglione, F. Corno; "Specifying Web Service Compositions on the Basis of Natural Language Requests"; ICSOC'05 3rd International Conference on Service Oriented Computing, Amsterdam, December 2005
24 Semantic Markup for Web Services (OWL-S), http://www.w3.org/Submission/OWL-S/
25 OMG, "Meta Object Facility V2.0", http://www.omg.org/spec/MOF/2.0
26 Web Service Semantics (WSDL-S), www.w3.org/Submission/WSDL-S/
27 Agarwal, V., Dasgupta, K., Karnik, N., Kumar, A., Kundu, A., Mittal, S., and Srivastava, B. A service creation environment based on end to end composition of Web services. In Proceedings of the 14th international Conference on World Wide Web (2005). ACM Press, New York, NY, 128-137.
28 Philippe Larvet, "Automatic Orchestration of Web Services Through Semantic Annotations", ICEIS 2007, 9th International Conference on Enterprise Information Systems, June 2007.
29 Jos de Bruijn, Christoph Bussler, Dieter Fensel, Michael Kifer, Jacek Kopecky, Rubn Lara, Eyal Oren, Axel Polleres, and Michael Stollberg. Web Services Modeling Ontology (WSMO). http://www.wsmo.org/
30 Yu, T., Kwei-Jay, L.: Service Selection Algorithms for Web-Services with End-to-End QoS Constraints. Information Systems and E-Business Management 3(2): 103-126 (2005)
31 L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q.Z. Sheng. Quality Driven Web Services Composition. In Proceedings of the 12th International Conference on the World Wide Web (WWW), Budapest, Hungary, May 2003. ACM Press.