A Feature-Oriented WSDL Extension for Describing Grid Services

Natalia Trejo, Sandra Casas, and Karim Hallar

Unidad Académica Río Gallegos, Universidad Nacional de la Patagonia Austral
Río Gallegos, Argentina
{nbtrejo,karimhallar}@gmail.com,
lis@uarg.unpa.edu.ar

Abstract. Grid computing and Feature-oriented Development Software are emerging technologies, which can be combined to analyze, model, and specify Grid services. In a Grid environment, there are a large number of similar resources provided by different parties, that may provide the same functionality, but different Quality of Service (QoS) measures. A feature-based approach is presented to optimize the development of Grid services and Grid service composition. WSDL specification is extended to contain useful description of both functional and non-functional characteristics by mean Design by Contract technique. In this way, Grid users can specify their QoS expectations and select suitable resources and use them for their Grid workflow at design time before its execution on the Grid.

Keywords: Grid service, Feature-oriented Software Development, Design by Contract, QoS Attributes, Grid Service Composition.

1 Introduction

This paper presents a unique blend of ideas from different technical areas: distributed computing, feature-oriented software development as new software engineering paradigm, service-oriented architecture, and software design methods. Grid technology [1] provides a distributed computing environment based on the aggregation and the sharing of comprehensive, safe and coordinated heterogeneous resources from different organizations dynamically pooled into Virtual Organizations (VOs).

Grid applications for service-based systems are usually not based on a single service, but are rather composed of several services working together in an application-specific manner to achieve an overall goal. An application developer has to decide which services offered by the Grid should be used in the application, and he/she has to specify the data and control flow between them. We refer to workflow as the automation of both - control and data flow - in Grid applications.

QoS is a "combination of several qualities or properties of a service" [2]. In the context of Service-oriented Architecture (SOA), it is a set of non-functional attributes that may influence the quality of the service provided by a Web service [3]. Usually,

F. Cipolla-Ficarra et al. (Eds.): HCITOCH 2011, LNCS 7546, pp. 64–72, 2012.

several Web/Grid services are able to execute a given task although with different levels of quality. In addition, different users or applications may have different expectations and requirements. However, workflows developers would have to offer multiple criteria related to non-functional or QoS characteristics. Thus, during design time of Grid workflows, it is important to consider non-functional attributes of the Grid application in order to satisfy the needs of each service requester/consumer before Grid workflow execution.

Feature-oriented software development (FOSD) [4] is a paradigm for designing and implementing applications based on features. A feature is an end-user visible characteristic or requirement in a software system. Software is modularized into feature modules that represent features [5]. To create an application, feature modules are composed. Thus, features can be composed in different combinations, e.g., omitting certain features or implementing alternative features. In this way, FOSD can be used to develop software product lines.

The concept of Design by Contract (DbC) was first introduced by Meyer [6] to facilitate component reuse. Grid services are components in computing paradigm based on Grid technology, and DbC can be used at the level of components specifying component contracts as part of the components interfaces including functional requirements and QoS restrictions of Grid applications based on Grid services.

In this work, an outline of the relevance of FOSD to Grid computing domain and how it could be useful in designing Grid services is given. Preliminary results from the combination of Grid computing with FOSD are introduced in order to represent Grid services including both functional and non-functional requirements on their representation.

The remainder of this paper is organized as follows. Section 2 briefly describes related concepts and works about Grid services composition and its QoS characteristics, FOSD approach and its application in Web services. Section 3 describes our model to represent and specify Grid service using FOSD approach and DbC to include functional and non-functional requirements. An XML-based language for contract to extend WSDL specification of Grid service is presented. Finally, the paper provides some conclusions and hints for future research.

2 Related Works

2.1 Grid Services Composition

OGSA standard (Open Grid Services Architecture) [7] addresses all the fundamental services of Grid computing such as job management, resource management, security services and service discovery. It specifies standard interfaces for these services and requires stateful services. Modern Grid middleware environments like Globus Toolkit (GT) [8], Unicore/GS [9] or gLite [10] are built on the Web Service Resource Framework (WSRF) [11] standard, which extends Web Services. This allows the creation of the so-called stateful Web Services that can store the state of operations and other properties without breaking the compatibility with standard Web services.

In WSRF, the Web service is described in a WSDL document and the resource is specified in a separate Resource Properties document. A WSDL description is an

XML document that contains all the information about service capabilities and invocation mechanisms. The capabilities are described in terms of the operations of the service and the input and output messages for each operation.

Composition process can be described as a process that implies the identification of functionalities required by the services to be composed and their interactions (e.g. control-flow or data-flow). Component services that are able to provide the required functionalities are then associated to services composition.

However, a WSDL document only addresses the functional aspects of a Grid service without containing any useful description of non-functional or QoS characteristic. Some high-level QoS dimensions have been identified as relevant for Grid services composition (time, cost, fidelity, reliability, security) [12]. If users were capable of specifying their QoS expectations of the workflow at design level, it would be possible to detect and avoid services incompatibility during Grid services composition. Therefore, in the selection of Grid services, Grid applications developers must consider both functional and QoS properties.

2.2 QoS Characteristics of Grid Services

Yu and Buyya [12] suggest that at the specification level, workflow languages need to allow users to express their QoS requirements. At the execution level, the workflow scheduling must be able to map the workflow onto Grid resources to meet users' QoS requirements.

In a Web/Grid environment, multiple Web/Grid services may provide similar functionalities with different non-functional property values. Therefore, all actors involved in workflow composition would have a mechanism to distinguish the best Web/Grid service according to functional and non-functional requirements.

According to the specification designed by the World Wide Consortium (W3C) [3], QoS requirements for Web services include the following attributes: performance, reliability, scalability, robustness, accuracy, integrity, accessibility, availability, interoperability and security.

The authors of [13, 14, 15] analyzed and proposed different solutions to represent QoS in Web services. Zeng et al. [13] proposed a model to evaluate QoS of both basic and composite services and a global service selection approach that uses linear programming techniques to compute optimal service execution plans for composite services. They present quality criteria in the context of elementary services (e.g. execution price, execution duration, reliability, and so on), which can be defined for an entire service or for individual service operations. The quality criteria to evaluate QoS of composite services are calculated based on QoS criteria of basic services. D'Ambrogio [14] introduced a lightweight WSDL extension for the description of QoS characteristics of a Web service. The WSDL extension, called Q-WSDL, is based on the OMG QoS and SPT Profiles and has been carried out as a meta-model transformation, according to principles and standards provided by Model Driven Architecture. In [15], the authors present a study of a Web service discovery system based on QoS and highlight the advantages and disadvantages of each system.

In the Grid computing field, the works of [16, 17, 18] present a solution to include non-functional requirements into Grid services. The work of [16] presents a framework for brokering of Grid resources which allows discovery and selection of

resources and automatic allocation of application tasks to them on the basis of both functional and QoS requirements. For this goal they extended ontology developed using OWL for QoS description for Web services. Acher et al. [17, 18] analyzed functional and non-functional variability of imaging services and proposed a Software Product Line Framework (SPLF). They addressed variability of Grid services for medical imaging by using an approach based on Software Product Lines. On the basis of meta-models handling functional and QoS variability, the SPLF describes possible types of services and workflows for the domain of medical imaging. It considers services variability, including a set of common properties and a set of possible differences. Thus, developers are able to describe the structure and the behavior of services, propose variants and define optional parts. Then, Grid workflows experts are able to transparently choose and deploy services from SPL and execute applications composed of several of them. End users just specify data and their requirements and QoS needs.

There are specific QoS aspects of Grid services beyond classical QoS attributes defined by W3C. These QoS requirements also depend on the nature of each Grid application, and could include attributes such as cost, reproducibility, predictability, minimum storage capability for storage services, user needs (e.g. emergency of computation, expected output quality, etc.).

2.3 Feature-Oriented Software Development

Feature-Oriented Software Development (FOSD) is a "paradigm for construction, customization, and synthesis of large-scale software systems where the main concept is the feature" [19]. These authors define a feature as a unit of functionality of a software system that satisfies a requirement, represents a design decision, and provides a potential configuration option. Software system is decomposed in terms of the features it provides. The concept of decomposition allows constructing well-structured software that can be tailored to the needs of the user and the application scenario.

From a set of features, many different software systems that share common features and differ in other features can be generated. The set of software systems generated from a set of features where they share common aspects as predicted variability, is also known as software product line [20].

Apel and Kaestner [19] also present a survey to convey the idea of FOSD as general development software. The concept of feature is used to structure the design and code of a software system. Features are the core units of reuse in this approach, and the variants of a software system vary in the features they provide. The software is generated in an efficient and correct way on the basis of a set of feature artifacts and a user's feature selection.

2.3.1 Feature-Based Approach to Develop SOA Applications

Apel et al. [21] present an approach that integrates the notions of services and feature-based product lines. The similarity between feature-based approaches and service-based approaches to software system construction is that both aim at structuring

complex software systems into manageable pieces. The authors also present the benefits of a feature-based approach to SOA and pose several challenges, particularly when services are black boxes implemented and deployed by different vendors. The vendors do not share code; only interface descriptions are available. They recommend creating a common feature model that is well defined for a domain. Based on this model, vendors can provide a feature-based specification for their services.

A few works for modeling SOA applications using features have been presented. In [22] a feature diagram notation is used to identify variability in Web Services architectures. However this approach focuses on the user's point of view instead of integrating Web services from multiple vendors. More recent works [17, 18] have analyzed variability of functional and non-functional requirements of medical imaging processing Grid services. The feature-based approach has been used to propose meta-models in order to handle functional variability and QoS mechanisms. Grid services are organized as product line architecture and feature models are used to structure relevant information in terms of service's variability. Family of services is defined as a set of concerns that exhibit variability, each being represented with several feature models. A set of composition operators is proposed to enable service composition.

3 Modeling Grid Services Using FOSD

Grid workflows may represent complex scientific and business processes, which normally change often. Therefore, firstly, we need to capture and represent each task of these processes by means of Grid services and their interfaces. We propose a new approach to model Grid services based on FOSD.

VOs usually share their resources using Grid services. These services are black boxes implemented and deployed by different organizations. Integrating off-the-shelf services located at different places and using interface descriptions generate Grid applications. DbC [6] is used at level of Grid services specifying contracts as part of the service interfaces. The contract will describe non-functional restrictions that Grid service must hold from its clients/service requester and vice versa.

In this way, a Grid service can offer interfaces that are detailed by Pre-conditions, Post-conditions and Invariant assertions of DbC technique, which could be related with input or output operations or non-functional features, e.g. QoS. We will use DbC in order to extend WSDL specification to functional and no-functional requirements. Non-functional requirements of Grid service composition will accomplish defining non-functional requirements for each Grid service.

3.1 Extending Grid Service Interface

Grid Contract Definition Language (GContractDL) is created in order to extend WSDL specification of Grid services using DbC technique. In this way, Grid developers are able to specify functional and non-functional restrictions during Grid service interface definition and Grid services composition.

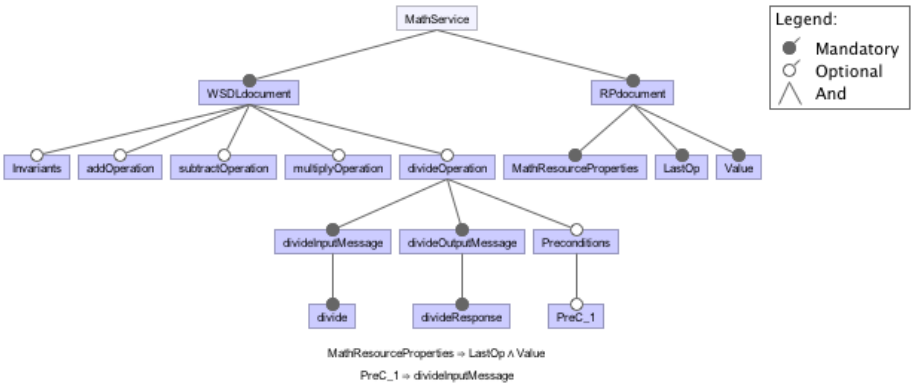

Fig. 1. An example of Grid service Feature Model

We give a simple example of Grid service, called MathService. This Grid service performs basic arithmetic operations using Resource Properties (RP), Math Resource Properties which contains two elements: LastOp (last operation) and Value. The internal logic of Math Service is as follows: Once a new resource is created, the Value RP is initialized to zero, and the LastOp RP is initialized to NONE. The elementary operations expect only one integer parameter. This parameter is added, subtracted, etc. to the Value RP, and the LastOp RP is changed to ADDITION, SUBTRACTION, PRODUCT or DIVISION accordingly. Also, the operations do not return anything. Suppose that a client or other service requests the division operation to MathService, then the feature model will be as shown in Fig. 1. Feature structure tree notation [23] is used to represent feature model of MathService.

Based on this MathService feature model, we can see it would be much better if we knew something more about the Grid service, e.g., that division operation does not accept zero value as input or that all operations cannot accept float data type as input values. We also could use this representation to define quality criteria at design level of workflow creation, such as service performance, result accuracy, data confidentiality, time, cost, fidelity, reliability and so on. All these non-functional requirements can be defined into the XML Schema used to produce and validate an extension of WSDL document of Grid services, as it will be seen in the next section.

3.2 GContractDL XML Schema

Figure 2 graphically represents the XML Schema of GContractDL. The root element is called `<contract>` which type is contractType (Fig. 2a), this element contains several occurrences of the elements `<operation>` and optionally several occurrences of `invariant` elements. The `operation` element describes each service operation defined on WSDL specification. Inside the `operation` element we can specify information about parameters, pre-condition and post-condition assertions. `Invariant` elements could be used to state non-functional aspects of Grid service, which must be satisfied before and during Grid service and workflow execution.

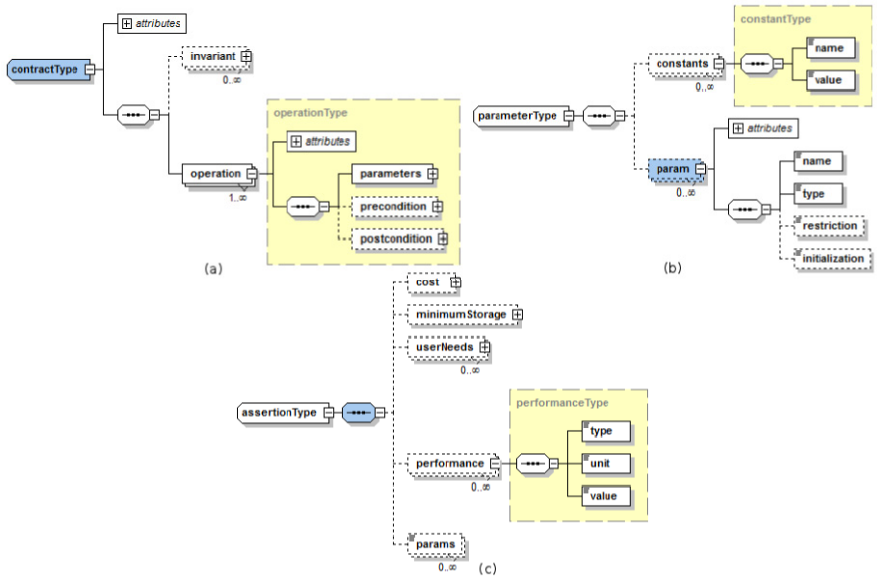

Fig. 2. Graphical representation of XML schema for GContractDL

The `parameter` element (Fig. 2b) describes operation parameters and return values, as well as constants that a service supports. For each parameter it is possible to specify ID, direction, and whether the parameter is required or optional. For each `parameter` element it is possible to define name, type, restriction and initialization.

`Pre-conditions`, `post-conditions` and `invariants` share the same structure (Fig. 2c). `Pre-conditions` are linked to operations and determine obligations of a client or service requester. An operation is guaranteed to work correctly if and only if pre-condition is satisfied. `Post-condition` describes what an operation guarantees, if pre-condition holds. `Invariants` are properties that must hold before, during and after Grid service execution. The child elements describing pre-conditions, invariants and post-conditions are related to non-functional attributes, such as `performance`, `cost`, `minimum storage`, and so on. `Params` element allows for specification of conditions for parameters, be it pre-conditions for input parameters or guarantees (post-conditions) for output parameters (results).

Using GContractDL we can extend WSDL specification by defining pre/post-condition assertion related to each Grid service operation and invariant assertion for each Grid service and which allows defining the whole workflow.

GContractDL can be applied at different levels of granularity. For each operation offered by Grid service, requirements to accomplish correct Grid service execution can be defined. Also, Grid developers are able to define results that service operation guarantees when these requirements are met. Moreover, during service composition, global conditions can be provided to the execution of each service through the invariants. In this way, DbC will ensure QoS of the entire Grid application.

4 Conclusion and Future Work

Grid services capabilities, as an extension of Web service, are expressed in XML by using WSDL. Unfortunately, a WSDL description only addresses the functional aspects of a Web/Grid service without containing any useful description of non-functional or QoS characteristics.

This paper has proposed a novel approach to combine FOSD and Grid computing in Grid service representation. We have used the first two FOSD phases to describe Grid service. Grid service functionality and non-functional attributes have been represented by a set of features. WSDL specification has been extended to support DbC elements in order to describe the behavior offered and required for a Grid service and include non-functional requirements. This is accomplished by means of XML-based language for Grid services contracts.

FOSD has several open issues related to their phases. Particularly, in the phase of *domain design and specification*, there has not been much work. Feature interaction occurs when the integration of two features modifies the behavior of one or both features in an undesirable way. *Feature Interaction Problem (FIP)* is still an open and hard research challenge [19] and is an issue wherever independently developed software components are required to work together. Furthermore, Calder et al. [24] suggests the needs are semantic specifications besides interface specifications because these are insufficient and hence feature interactions would be an issue between Web services and also between Grid services. Grid service specification must be improved to add more behavioral information and test algorithms to detect FIP among Grid services based on pre/post-conditions and invariant elements of DbC.

As future work we plan to define a feature interaction taxonomy, which would allow detecting undesirable interaction when Grid services are composed. Also we will design a prototype of a notation to specify Grid workflow on the basis of feature-based WSDL extension and that allows feature interaction detection at design-level.

References

1. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the Grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications 15(3), 200–222 (2001)
2. Menascé, D.: QoS issues in web services. IEEE Internet Computing 6, 72–75 (2002)
3. W3C Consortium: QoS for Web Services: Requirements and Possible Approaches (2003), http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/
4. Kaestner, C., Thum, T., Saake, G., Feigenspan, J., Leich, T., Wielgorz, F., Apel, S.: FeatureIDE: A tool framework for feature-oriented software development. In: 31st International Conference on Software Engineering, pp. 611–614. IEEE Computer Society, Washington, DC (2009)
5. Prehofer, C.: Feature-Oriented Programming: A Fresh Look at Objects. In: Aksit, M., Auletta, V. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 419–443. Springer, Heidelberg (1997)
6. Meyer, B.: Applying design by contract. Computer 25(10), 40–51 (1992)

7. Foster, I., Kishimoto, H., Savva, A., Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J., Reich, J.: The Open Grid Services Architecture, Version 1.0, http://www.gridforum.org/documents/GWD-I-E/GFD-I.030.pdf
8. Globus Alliance: Globus Toolkit, http://www.globus.org/toolkit/
9. Almond, J., Snelling, D.: UNICORE: uniform access to supercomputing as an element of electronic commerce. Future Generation Computer Systems 15, 539–548 (1999)
10. EGEE Project: gLite, http://glite.web.cern.ch/glite/
11. OASIS: Web Services Resource Framework (WSRF) 1.2, http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf
12. Yu, J., Buyya, R.: A taxonomy of workflow management systems for Grid computing. Grid Computing 3(3-4), 171–200 (2005)
13. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.: Quality driven web services composition. In: 12th International Conference on World Wide Web, pp. 411–421. ACM, New York (2003)
14. D'Ambrogio, A.: A model-driven WSDL extension for describing the QoS of Web services. In: IEEE International Conference on Web Services, pp. 789–796. IEEE Computer Society, Washington, DC (2006)
15. Rajendran, T., Balasubramanie, P.: Analysis on the Study of QoS-Aware Web Services Discovery. Journal of Computing 1(1), 119–130 (2009)
16. Ranaldo, N., Zimeo, E.: A framework for QoS-based resource brokering in grid computing. In: WEWST (2008)
17. Acher, M., Collet, P., Lahire, P., Montagnat, J.: Imaging services on the grid as a product line: Requirements and architecture. In: 12th International Conference, pp. 137–142. Lero Int. Science Centre, University of Limerick, Ireland (2008)
18. Acher, M., Collet, P., Lahire, P., France, R.: Managing Variability in Workflow with Feature Model Composition Operators. In: Baudry, B., Wohlstadter, E. (eds.) SC 2010. LNCS, vol. 6144, pp. 17–33. Springer, Heidelberg (2010)
19. Apel, S., Kaestner, C.: An overview of feature-oriented software development. Journal of Object Technology 8(4), 1–3 (2009)
20. Weiss, D., Lai, C.: Software Product-Line Engineering: A Family-Based Software Development Process. Addison-Wesley (1999)
21. Apel, S., Kaestner, C., Lengauer, C.: Research challenges in the tension between features and services. In: 2nd International Workshop on Systems Development in SOA Environments, pp. 53–58. ACM, NY (2008)
22. Robak, S., Franczyk, B.: Modeling Web Services Variability with Feature Diagrams. In: Chaudhri, A.B., Jeckle, M., Rahm, E., Unland, R. (eds.) NODe-WS 2002. LNCS, vol. 2593, pp. 120–128. Springer, Heidelberg (2003)
23. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, S.: Feature-Oriented Domain Analysis (FODA). Feasibility Study. Tech. Rep. CMU/SEI-90-TR-21, ESD-90-TR-222, Software Engineering Institute, Carnegie Mellon University (1990)
24. Calder, M., Kolberg, M., Magill, E.H., Reiff-Marganiec, S.: Feature interaction: a critical review and considered forecast. Computer Networks: The International Journal of Computer and Telecommunications Networking 41(1), 115–141 (2003)