Context Modeling and Constraints Binding in Web Service Business Processes

Kosala Yapa Bandara
Dublin City University
Dublin 9, Ireland
kyapa@computing.dcu.ie

MingXue Wang
Dublin City University
Dublin 9, Ireland
mwang@computing.dcu.ie

Claus Pahl
Dublin City University
Dublin 9, Ireland
cpahl@computing.dcu.ie

ABSTRACT

Context awareness is a principle used in pervasive services applications to enhance their flexibility and adaptability to changing conditions and dynamic environments. Ontologies provide a suitable framework for context modeling and reasoning. We develop a context model for executable business processes – captured as an ontology for the web services domain. A web service description is attached to a service context profile, which is bound to the context ontology. Context instances can be generated dynamically at services runtime and are bound to context constraint services. Constraint services facilitate both setting up constraint properties and constraint checkers, which determine the dynamic validity of context instances. Data collectors focus on capturing context instances. Runtime integration of both constraint services and data collectors permit the business process to achieve dynamic business goals.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*Domain-specific architectures, Languages*

General Terms

Design

Keywords

Web Service, Semantics, Context, Ontology, Constraints

1. INTRODUCTION

Web services provide a platform for bridging heterogeneous applications. The Web services platform has the capability of enabling high-level processes referred to as composite web services. Composition addresses the situation in which a user request cannot be satisfied by a single Web service, whereas a composite web service consisting of a combination of web services could satisfy this request. The Business Process Execution Language for Web Services (WS-BPEL or BPEL for short) is the de facto standard for services composition. Processes are executed by a BPEL engine, which creates a process instance when an invocation activity (receive) is triggered and terminates the instance after completing the corresponding reply activity. The process instance interacts with other services (partners) through further invocation activities.

Context has been investigated in many web services research projects [5, 10, 9, 8]. Context awareness is a major enabler in ubiquitous computing and offers promising ways to adapt and personalize applications [4]. The main objective of our work is to facilitate the development and deployment of context determined dynamic web services applications. Standard web service descriptions lack context information support. However in context determined dynamic web services applications, runtime context constraints binding needs to be integrated [2].

We can identify the three central benefits of context constraint modeling and dynamic integration in business processes: Firstly, context and constraint services binding support a business process to achieve dynamic business goals. Secondly, integration of various constraints such as technical constraints and business constraints, based on the context ontology as the information integration platform. Thirdly, enhanced modeling of constraints and the reasoning could be achieved through the context ontology. Our approach creates context instances dynamically and binds context instances to constraint services within the executable business process to achieve dynamic business goals. Moreover, our context ontology could be used as an information integration platform and also supports enhanced modeling of constraints and reasoning.

In this paper, Section 2 presents our ontology-based context constraints framework. Then, we discuss in Section 3 our web services context ontology illustrating context categories. In section 4, we discuss the operationalisation of context constraints. Related work is discussed in Section 5 and some conclusions are presented in Section 6.

2. ONTOLOGY-BASED CONTEXT FRAMEWORK FOR SERVICE PROCESSES

While some authors have investigated comprehensive service and process ontologies [5, 10, 9], our aim the investigation of context aspects that are relevant for the execution and in particular the dynamic composition and monitoring of processes. Therefore, the operationalisation of context element is a critical factor here. This process of operational-

CASTA'09, August 24, 2009, Amsterdam, The Netherlands.

ising context aspects is based on constraints determination, constraints specification and supports business process instrumentation with constraints using a weaving technique – see Fig. 1. Execution-relevant and measurable constraints are bound to the process execution. Context constraints determination is supported by service level agreements.

The autonomic composition of web services usually starts with a planning process based on an abstract goal. An abstract plan is produced, from which an executable service process (e.g. in WS-BPEL) can be derived. This forms the starting point for the integration of context constraints. While functional aspects (also part of the context in our case) can be considered at planning stage, some constraints only be needed during the execution. We present a context ontology capturing all aspects of a service in relation to its dynamic execution environment. Context constraints to be integrated are generated based on service level agreements and linked to constraint services (constraints binding).

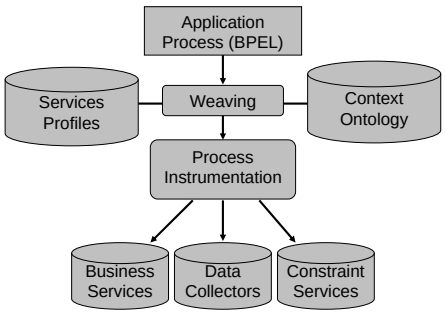

Figure 1: Context Constraints Weaving Framework

3. CONTEXT ONTOLOGY MODEL

Context has been recently explored to facilitate the development and deployment of context-aware and adaptable Web services [8]. A modeling context for pervasive computing environments, identifying location, user, activity and computational entity as fundamental context categories is defined by [11]. Often, location is the central context concern in these platform-centric approaches, while others [5, 10] go beyond our focus on dynamic aspects.

3.1 Context Model Determination

In order to determine a comprehensive context model for dynamic services composition, we followed an empirical approach by looking at three case studies. First, online billing and payment for utility services gives users the flexibility to use different types of devices and payments in different types of currencies [2]. Second, an e-learning courseware generation scenario reflects knowledge and information intensive applications where the course content is automatically generated based on information and knowledge resources. Third, a multilingual convenience service scenario reflects a service-based application in modern information, communications and social networking environments. These case study scenarios – which are all based on real software systems – have been defined in three different domains to illustrate needs of a targeted context model.

From these we derived a context model ontology. Four context categories are identified in the proposed context ontology as Functional Context, which is useful in autonomic services composition in general, Quality of Service Context, which is useful in achieving dynamic composition, Semantic and Domain Context, which is useful in achieving autonomic composition in different organizations and the Platform Context which reflects the technology infrastructure.

We have considered more comprehensive business service and process context models in order to validate the identified context aspects [5, 10, 9]. Our context ontology focus is on the immediate layer (according to Rosemann et al.) and the functional and input/output aspects (according to Heravizadeh et al.). Aspects such as settlement or payment (O'Sullivan et al.) or the outer layers of Rosemann et al.'s onion model do not apply as they are not execution-related and therefore not dynamically relevant. The definition of service quality supplied by O'Sullivan et al. (measure of the difference between expected and actual service provision) applies here. We therefore include functional aspects into our context model as these might vary dynamically.

3.2 Web Services Context Ontology

Based on our case study observations, we define dynamic context as client, provider or service related information, which enables or enhances efficient integration among clients, providers and services [2]. Services must be aware of their context in order to be able to automatically adapt to changing context. Choosing an ontology format provides a common vocabulary, which is machine-readable, to share context information relating to the Web services domain. A number of individual context categories are defined in the context ontology. These are grouped into four top-level context areas. The identified context categories are:

Functional Context: This describes the operational features of services. The notion of functional context in Web services is sub-grouped into Syntax, Effect and Protocol:

- *Syntax:* This includes the list of input/output parameters that define operations' messages, the data types of these parameters for invoking the Web service.
- *Effect:* Includes the pre-conditions and post-conditions, i.e. the operational effect of an operation execution.
- *Protocol Context:* Refers to a consistent exchange of messages among services involved in services composition to achieve their goals. The protocol context includes context on conversation rules and data flow.

Quality of Service Context (QoS): In service computing environments, a number of services often perform a similar function. The difference manifests itself in the form of different levels of quality. Facilitating end-to-end quality of service in autonomic Web service composition is a critical and significant challenge. One difficulty is to explain what quality is and how to determine good or bad quality. End-to-end QoS is critical for a business process, which is composed of both single and compound services [12]. Accepted definitions describe quality as "the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs" [7]. Stated needs are explicitly declared by the users and implied needs refer to requirements not known to users. Dynamically relevant qualitative properties can be organized into four groups based on the type of measurement performed by each attribute:

- *Runtime Attributes:* These attributes are measurable properties related to the execution of a service. Performance; the measurement of the time behavior of services in terms of response time, throughput etc. Reliability; the ability of a service to be executed within the maximum expected time frame. Availability; the probability that a service is accessible.
- *Financial/Business Attributes:* These attributes allow the assessment of a service from a financial/ business perspective. Cost; the amount required for execution. Reputation; measures the service's trustworthiness. Regulatory; a measure of how well a service is aligned with government or organizational regulations.
- *Security Attributes:* These attributes describe whether the service is compliant with security requirements. Integrity; protecting information from being deleted or altered without permission. Authentication; ensure that both consumers and providers are identified and verified. Non-repudiation; the ability of the receiver to prove that the sender really did send the message. Confidentiality; protecting information from being read or copied by anyone not authorized.
- *Trust Attributes:* Attributes refer to establishment of trust relationships between client and provider, which is a combination of technical assertions (measurable and verifiable quality) and relationship-based factors (reputation, history of cooperation).

Other quality attributes or groups can be added to these fundamental groups without altering the generality.

Domain Context: Each application domain may need its own context (often called locale) for service interaction:

- Semantic: refers to semantic framework (i.e. concepts and their properties) in terms of vocabularies, taxonomies or ontologies.
- Linguistic: the language used to express queries, functionality and responses.
- Measures and Standards: refers to locally used standards for measurements, currencies, etc.

Platform Context: The technical environment a service is executed in:

- Device: refers to the computer/hardware platform on which the service is provided.
- Connectivity: refers to the network infrastructure used by the service to communicate.

4. OPERATIONALISATION OF CONTEXT CONSTRAINTS

The purpose of our context ontology is to capture context aspects that are relevant for dynamic service process composition and monitoring. We weave constraints for dynamic monitoring into processes [3, 2]. For each service, we

- determine the service profile (service interface and service level agreement-specific aspects),
- for each relevant aspect generate constraints using the context ontology and its instances
- depending on the context category, create an instrumentation binding, i.e. calls to data collectors and constraint services that compare observed and required context-based constraints,
- insert the instrumentation code into an abstract business process.

At the centre of the integration of constraint validation into a composed service process is a mapping. Attributes of the context model aspects are connected to concrete values at the instance level. These form the context constraints. Pairs of attributes with their associated values form abstract constraints. A preparation step for the final mapping is the determination of data collectors and data initialisers (e.g. SetBillFormat) that support the constraint condition.

A key observation here is that the implementation of constraint validation is context category-dependent:

- The Functional context details, e.g. parameters and protocol aspects. Pre/post-condition validation is used, but no data collectors.
- The Quality of Service constraints usually require data collectors to monitor variable quality properties before validating constraints.
- The Domain and Platform constraints refer to data collectors to determine environment conditions. In contrast to the quality monitors, these are static properties (such as language or device) that need to be queried, but not measured.

This category-dependency allows for uniform constraint monitoring within the categories, which is an advantage for efficient constraint integration.

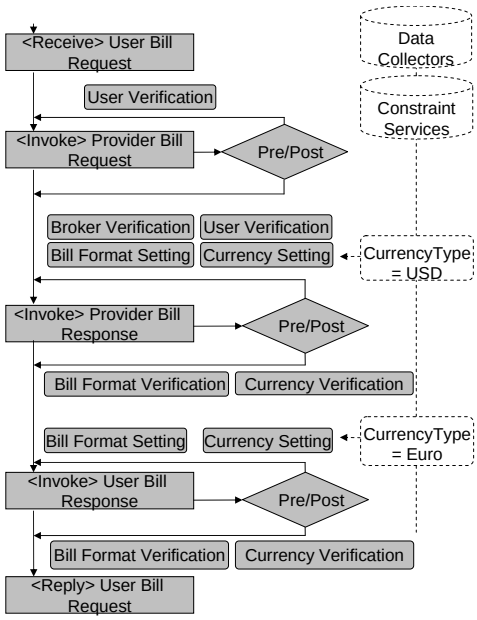

Figure 2: Constraints Integrated Process

In the example (Fig. 2), User and Service Broker agree on using different devices and currency types for utility bill payments during registration (service level agreements). The

user request is the starting point for generating context constraints based on the agreements. The context constraints represent context instances (generated based on the context ontology). Pre/post constraint services and data collectors are bound to the process. The *Bill Format Verification* constraint service checks the appropriate bill format for each display device determined by the context instance (Device Context). Similarly, the bill amount is calculated and set for the requested currency (Measures and Standards Context). When a user request arrives (<`receive`>) from a mobile device, the Display Bill Format is determined. Then, the appropriate constraint service is determined. This derived context is bound to a constraint setting service (*BillFormatSetting*). All constraint instrumentations are integrated into the BPEL process, which is a sequence of invocation of services (<`invoke`>). For the UserBillResponse service, the following constraint instrumentation is generated:

- Pre: initialise data collectors with the required settings for format and currency,
- Post: verify the pre-execution settings against the actual delivered results of the service

Another example is response time, where pre- and post-execution time stamps are compared.

5. RELATED WORK

The related work in this area covers previous work on context modeling and constraints used in dynamic service architectures. Broens proposes a realization of a context binding infrastructure called the Context-Aware Component Infrastructure (CACI). This realizes context-binding transparency and is composed of a context binding mechanism and a context discovery interoperability mechanism [4]. However, this approach is for component-based application developments and not specific for service process composition. Hong et al. present a context-aware manager-based architecture to support user-centric ubiquitous learning services, and describe an ontology-based context model for intelligent school spaces [6]. This does not provide a solution for constraint integration in service compositions. Medjahed et al. propose a context-based matching approach, which has detailed information about context information and context policies, but does not address dynamic context constraints binding [7]. The METEOR-S project focuses on constraints-driven services composition [1]. It distinguishes data, functional and quality of service semantics, but semantics would need to be further investigated. Our approach is more general approach in capturing and operationalising the semantics of Web services using the context model ontology with context instances and constraint services.

6. CONCLUSIONS

We have introduced an ontology-based context framework for dynamic web service processes to achieve dynamic business goals. We defined a notion of context for executable Web service processes formalized as a context ontology. The operationalisation of context dynamically as verifiable constraints is the aim. This requires the integration of constraint handling and the binding of the process elements to additional constraint-verifying services.

The major contributions defined in our approach are an ontology-based context model to support service processes with dynamically monitored business goals; the integration of context constraints such as technical and business constraints based on a context ontology as the information integration platform and category-dependent uniform constraint integration technique.

Compositionality needs to be further addressed in two forms. Firstly, the service process composition and its effect on service-specific constraints within the restrictions of the context ontology need to be discussed further. Secondly, the compositionality of constraints – to determine inconsistencies in context constraints – is another open problem.

7. ACKNOWLEDGEMENT

This work is supported by the Science Foundation Ireland through the CASCAR project.

8. REFERENCES

[1] R. Aggarwal, K. Verma, J. Miller, and W. Milnor. Constraint driven web service composition in meteor-s. In *Proceedings of the IEEE International Conference on Services Computing*, 2004.

[2] K. Y. Bandara, M. Wang, and P. C. Dynamic integration of context model constraints in web service processes. In *International Conference on Software Engineering*. IASTED, 2009.

[3] L. Baresi and S. Guinea. *Towards Dynamic Monitoring of WS-BPEL Processes*, volume 3826 of *Lecture Notes in Computer Science*. Springer, 2005.

[4] T. Broens. *Dynamic context bindings - Infrastructural support for context-aware applications*. PhD thesis, Univ. of Twente, 2008.

[5] M. Heravizadeh, J. Mendling, and M. Rosemann. Dimensions of business processes quality (qobp). In *Proc. of the 6th International Conference on Business Process Management Workshops (BPM Workshops 2008)*, 2008.

[6] M. Hong and D. Cho. Ontology context model for context aware learning service in ubiquitous learning environments. *International Journal of Comp.*, 2008.

[7] B. Medjahed and Y. Atif. Context-based matching for web service composition. *Distributed and Parallel Databases*, 21:5–37, 2007.

[8] M. Mrissa, C. Ghedira, D. Benslimane, and Z. Maamar. *Context and Semantic Composition of Web Services*, volume 4080 of *Lecture Notes in Computer Science*. Springer, 2006.

[9] J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. What's in a service? *Distributed and Parallel Databases*, 12(2/3):117–133, 2002.

[10] M. Rosemann, J. Recker, and C. Flender. Contextualization of business processes. *International Journal of Business Process Integration and Management*, 3(1):47–60, 2008.

[11] X. Wang, D. Q. Zhang, T. Gu, and H. Pung. Ontology based context modeling and reasoning using owl. In *Proceedings of the Second Annual Conference on Pervasive Computing and Communications Workshops*. IEEE, 2004.

[12] Z. Wu and J. Luo. Qos-resource graph model for web service composition in service oriented computing. The Sixth International Conference on Grid and Cooperative Computing, IEEE, 2007.