CASIA: A Context-aware Approach for Service Identification aligned to Business

Michel Antunes da Silva[1], Flávia Maria Santoro[1,2] and Leonardo Guerreiro Azevedo[1,2]

[1] Depto de Informática Aplicada – Universidade Federal do Estado do Rio de Janeiro Rio de Janeiro, Brazil
[2] Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec) – UNIRIO, Rio de Janeiro, Brazil

Abstract. Organizations use services to support the implementation of their business process. Service requirements are identified at design time and developed for the application that implements the business process. This makes the process highly dependent on the service that supports it. In addition, due to changes in the business process, the supporting service might no longer adhere to business process at any time. Thus, it is needed to adapt the application that implements the business process to consume another service that best fits its needs. This paper describes an approach for flexible identification (discovery) of services available in service repositories according to the business context. This approach aims at discovering the most appropriate service to meet business requirements.

1 Introduction

The Service Oriented Architecture (SOA) is considered by [10] as the computing paradigm that uses services as fundamental elements for developing applications. Services are pieces of self-contained features that have exposed interfaces, invoked via messages [8]. The development of services is directly linked to business process in order to give agility to the development lifecycle [15]. So, services should expose the features relevant to the business.

Companies are likely to suffer continuous changes in their environments, where systems are available through computers and network, as well as the availability of services from any service supplier. Therefore, companies need to be flexible to adapt to changing aspects of the environment without losing its compliance with immutable aspects considered stable [12]. However, we have not identified consolidated proposals for dynamic discovery of services to be consumed by applications that support a particular business implementation at the time this adaption is required. Besides, [10] presents that the main challenge of service discovery is using automated means to accurately discover services with minimal user involvement. This requires to make explicit the semantics of both the service provider and requester.

[14] indicate that context information may help in service adaptation to changes. According to [1], context is defined as constraints that limits something without in-

Antunes da Silva M., Santoro F. and Guerreiro Azevedo L. (2010).
CASIA: A Context-Aware Approach for Service Identification Aligned to Business.
In *Proceedings of the International Joint Workshop on Technologies for Context-Aware Business Process Management, Advanced Enterprise Architecture and Repositories and Recent Trends in SOA Based Information Systems*, pages 73-81
DOI: 10.5220/0003024900730081
Copyright © SciTePress

tervening in it explicitly, *i.e.*, any information that can be used to characterize the situation of an actor, which may be a person, place or object. Thus, we argue that it is necessary improving the contextual characteristics shared between business process and services. Increasing the capacity to understand service context will make possible to automate service discovery in equivalent contexts.

The goal of this paper is to describe an approach which structures the discovery of services aware of business context. It includes a framework that considers context information for alignment between business process and service. Besides, we propose the use of the aspect-oriented programming paradigm [5] to capture business context in order to support the discovery of the service in service repositories. Therefore, the proposed framework allows the application to use a new service (the better choice) without code maintenance.

The paper is organized as follows: Section 2 discusses related work; Section 3 presents the proposal; Section 4 described an application scenario; and Section 5 concludes it, pointing out future work.

2 Related Work

Companies need to increase the agility to respond to changes according to new market needs and challenges. Consequently, recent works aim at developing mechanisms to combine business perspectives with available software and hardware assets.

[4] present a framework for adaptable and context sensible web services. This framework uses the header of web service SOAP[1] message to transmit context information to a service repository. The focus of this work is in the automatic and transparent processing of context and in the extension of context types without considering how context is stored by service consumers. However, [4] concerns are for mobile computing requirements, and the main context attribute is the position of service consumer. Therefore, this work does not consider the business process that is supported by the service.

[11] propose to separate the logic implemented in a service from context features using a model-driven approach. Their proposal is based on an extension of the ContextUML metamodel [13] and generates code in an aspectual programming language. Therefore, the logic and context separation is defined in models and, then, passed to generated code. The use of an aspectual language allows handling the contextual layer in design and execution time. However, this work presents only a solution for the implementation for automation of code generation, not considering questions about which attributes are required for service identification aligned to business process.

[9] proposes to separate the elements concerning web services by categories, such as: organizational arrangements, service types, user characteristics, user state, transaction history, resource state and transaction state. The organizational arrangements include the organizational structure of the company, the relationship between people

[1] *Simple Object Access Protocol (SOAP) is a protocol used* for *message exchange in* a distributed platform based on *XML*.

and people and organizations, current policies, contractual agreements and current partnership. Service characteristics include all static elements of service environments (all available services, service registry, brokers). User characteristics are related to the long term attributes in a possible service consumer, such as preferences and limitations, level of expertise etc. The state of the user includes his geographic position in a specific time, physical characteristics and the area around him. The transaction history is the log of executed transactions. The state of resources is related to information about technical resources (computers, network etc) in the web service environment. The transaction state corresponds to information about the user's current transaction, such as, identity provider, position of services in a composed services, the current step in a composed service, the passed time after an operation initialization etc.

[9] considers that service discovery and selection may depend on all proposed categories, except the transaction state. However, [9] proposes general guidelines and challenges about architectural issues concerning web services contextualization, for example, context representation, security and privacy issues, scalability, service grouping and context information handling etc. He presents important categories for context definition, but his proposal do not address service discovery using these categories.

[3] propose the use of technical process semantic and business process semantic of a registered service for service discovery. The semantic of technical process is the service scope, while semantic of business process is related to company business domain. They suggest automatic creation of process context by business process as soon as a function requires a web service execution. They present guidelines to structure process context related to business scope and characteristics to discover best fit services. Besides, they consider that a business model is composed by a domain that includes business rules and roles. Process context is composed by a set of criteria and a weight vector. Criteria are defined as properties used to improve service discovery precision and the vector of weights (in a range of 0 to 1) is used to sort the services. The weight vector is defined by web service input and output parameters related to a set of priorities and a set of Quality of Services (QoS) attributes, such as, latency, cost, confidence etc. An example of criterion used to discover a service, according to service consumer needs, is find a service that provides alkaline battery delivery, because for the consumer alkaline batteries has longer lifetime and are important for his business. High priorities could be related to a common domain for all providers (defined as static[2] priority) and to domains of all services already found by other roles (defined as dynamic[3] priority).

Besides, [3] proposes to extend the Universal Description, Discovery and Integration (UDDI) in order to support aware comparison of business context. However, [3] conclude that his proposal uses a simple process context model in order to simplify the research and improvements are required using OWL-S to increase the semantic of the solution. Their proposal is similar to our proposal; however, they do not present a contextual representation that guarantees the alignment of business process and sup-

2 Static priority is defined as the same way to discover service for different roles in a business process.

3 Dynamic priority is defined as the way that must be different to discover the service for different roles in a business process.

porting services. For instance, their proposal does not consider the context of the user that started business process execution.

There are several researches searching for an ideal solution to achieve alignment between business processes and services, and the use of this alignment information for service discovery. However, there are not yet conclusive definitions around that alignment. In the next section, we present a proposal to address those problems.

3 CASIA: Context-aware Service Identification aligned to Business

Conventional UDDI server provides a service discovery through the pair: key and value [3]. However, UDDI does not provide enough mechanisms to handle complex discovery requests for service, *e.g.*, Is the service *S'* supports the activity *A'* that is running on the business process scope?; What are the services to support the activity *A'* that is performing by the official *F'*? Therefore, in our approach, the UDDI is adapted to ensure that the service will be available together with its proper context, making it possible to find out the best service to fulfill the request of a process.

The goal of our proposal is to ensure the discovery of appropriate service to support the process during execution. We aim at improving the quality of the comparison between process context and service context for service identification. Therefore, one requirement is to specify the relevant contextual elements for process context and service context definition and the relationship between them. Besides, it is required an efficient and not intrusive mechanism to intercept an issue for a service, during process execution, in order to start service discovery.

The contextual elements are designed and grouped based on categories proposed in [9]. The semantics proposed in [3] were analyzed according to the categories of [9], in order to identify deficiencies and improve it.

[3] stated that the OWL-S (Ontology Web Language for Services) is able to express service context in the most effective way. OWL-S is a language for describing services that provides a standard vocabulary to be used in conjunction with other aspects of the OWL language description (Web Ontology Language) in order to create service descriptions [9]. As a result, Ontology is adopted as a model to represent context and moreover, it was needed to enhance existing models to meet the inherent requirements in the contextual alignment between processes and services.

A framework was established aiming to appropriate service discovery, semantically contextualized, to meet the demand of the process. It was necessary to guarantee flexibility in the capture of the contextual elements inherent to business process and, after that, make the comparison with the service context. Therefore, an AOP language was chosen as a non-intrusive form of implementation. The language adopted was the *AspectJ*, because this implementation has the static and dynamic aspectual weaving feature, *i.e.*, the combination at compile time and execution time [7].

The dynamic weaving enables the aspectual module to deal with the service discovery in a modular way, independent from the implementation that supports process execution. This ensures low coupling and transparency between the business process and the layer that handles its context. So it allows changes to incorporate the process context in the process execution instance without suspending it.

The logical architectural model is presented in Figure 1, in order to clarify the characteristics inherent in this research proposal, and is described as follows.

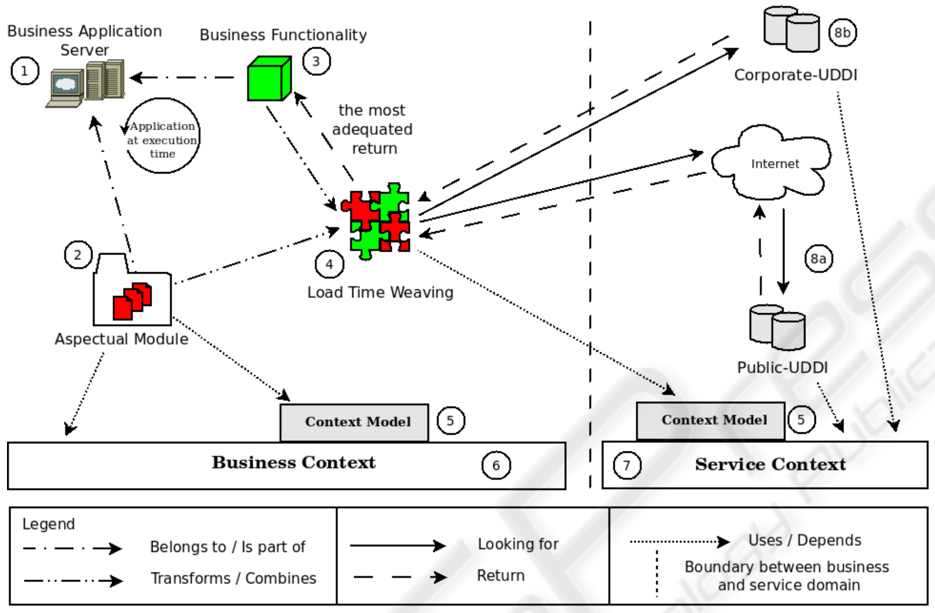

Fig 1. The logical model of the architecture.

1. Business Application Server: It is the execution environment of an application that implements business processes of the company. It is where the services, which best meet the business functionality, run.
2. The Aspectual Module: This component is responsible to centralize the logic of business context's handling and logic of service discovery. This component uses a context model to make a semantic discovery of services.
3. Business Functionality: It represents a functionality (implementation of a requirement) that is supported by a service in the instance of a given business process.
4. Aspectual Weaving: In this module, the concern implemented in the Aspectual Module is unified to a specific point in the instance of business process; thus, providing additional behavior to this instance.
5. Context Model: The information stored according to this model describes business context and service context (as an Ontology), and is used to help the discovering of the most appropriate service to meet the needs of the process instance.
6. Business Context: Repository of data describing business context according to the context model.
7. Service Context: Repository of data describing service context according to the context model.
8. Service Repositories: They comprehend the environments available on the Internet where services are offered for consumption. Services can be consumed through a

contract between provider and consumer or they can be consumed for free. The repositories can be divided into:

- 8a. Public Service Repositories: These include repositories that offer public services to be consumed.
- 8b. Corporate Service Repositories: These include service repositories within the company where the services are offered.

The operation of this architecture, when available in the corporate environment, will have its beginning when the implementation of business functionality starts in a specific need (the target). When a specific part of the business issue a service, then the aspectual module intercept this need to find out the most suitable service to support that business process.

In order to obtain this service, a context model which represents the semantic elements of the business and service is used. It increases the possibilities to meet the real alignment between process and service.

To select one service to support the process, it is necessary to group and rank the services identified, and then, the most suitable service is found out according to the contextual model and is returned to the original instance of the process.

4 Application Scenario

This section discusses a scenario that exemplifies the use of our approach within the proposed architecture.

4.1 Description

The scenario is related to the Sales Process that works online in a bookstore. In this case, the Company has no physical stock. Thus, there is a need to control and optimize the on-demand logistic process, which requires lots of information. However this information is not available in the execution environment, *i.e.*, the customer data are not easily recoverable.

In order to obtain the correct business context, user's information and company's information have to be registered into business data repositories. This information helps finding the adequate service to business process. Thus, the supporting system has to identify users, *i.e*, customers that participate in this business process, which is a precondition.

The process starts when customers want to buy a book. They search for a new book in the website and select it to pay. After that, the system will generate the invoice. Subsequently, customers make the payment, and the order will be delivered to them, according to the informed delivery address.

Nevertheless, logistic process will be performed by services, *i.e.*, services will support the purchase process and the delivery process. These services are normally provided by bookstores and carrier companies.

Therefore, when customers start the payment activity, the system will be aware of the business context to identify adequate services to get the requested book and to deliver it to customer address. These services will be selected according to customers' necessities and contextual conditions of the company.

At the end of the process, the customer will be informed about an estimated delivery date and the invoice. After that, the customer will pay the order. Then, the order will be delivered to customer address and the process will be finished.

4.2 Application of the Proposed Architecture

The proposed architecture works identifying and finding the service to rightly support business process. Business process is executed within Business Application Server (Fig1-1). The process is composed by functionalities (business functionalities).

When business functionality (Fig1-3) tries to invoke a service, the aspectual module (Fig1-2) intercepts this event at runtime (Fig1-4) and evaluates if the invoked service is the most adequate to support this business process specific execution. Therefore when, *e.g.*, customers finish the order at the payment page, the aspectual module intercepts invocation and evaluates if the service, which was specified at design time, is still adequate, according to business context involved on this execution.

The aspectual module obtains business context (Fig1-6) from business environment and uses it to identify a group of service repositories (Fig1-8) that are suitable to support business process. The aspectual module uses business context to find services. It searches service repositories with business context to find candidate services, *i.e.*, services that might carry out that business process.

Services are selected based in their contextual descriptions identified as service context (Fig1-7). These descriptions are organized using a context model (Fig1-5) which is a semantic model for both the service and the business process.

The aspectual module uses both context models to rank and select the more adequate service to be consumed by business process. This ranking is performed according to common rules for both contexts. After that, when the best service is selected, the aspectual module invokes it, and it returns a response message to business process execution. The semantic of the response message must be known, because the business process will use it in its execution. In this scenario, the first service invocation occurs after the aspectual module intercepts the purchase method for book request using a supplier service. In order to identify the supplier, the context of purchase method of all suppliers services are analyzed, consistent with customers preferences and situations, and Company arrangements, *e.g.*, cost, time to delivery, product's availability, service's availability, contractual fidelity. Afterwards, it is time to organize candidate services.

For this reason, the ranking is tidy based on QoS (Quality of Service) values, *e.g.*, time to response, ensure compliance with service level agreement, confidence rate of the response message semantic and so on. Finally, one of those services is elected to support business process. Even when there is only one candidate service to be ranked, this service context must be analyzed to evaluate if it can support the business process in a minimum level of QoS.

As an example of service ranking, the purchase service identified by the aspectual module could be from Amazon, Barnes&Noble and Booksite bookstores. Nevertheless the Booksite´s purchase service has not a good “time to response” (time required by the service to process the request and return an answer) and Amazon bookstore has not an adequate ´time to delivery´ (time to ship the book to the customer).

Then it invokes Barnes&Noble's purchase service (the elected candidate). However, the response could not match the response expected by business process. In order to succeed in pursuing execution of business process, the aspectual module handles this object returned to adapt it semantically to the expected format.

The second service invocation is performed by the delivery activity, and a flow similar to the purchase one is executed.

4.3 Discussion

Our proposal has many benefits. The first one is related to the flexibility to deal with different business domain; since it work at runtime level and no changes is needed in business process flow, as well in application code to invoke different services. The aspectual module features can be customized without impact the business execution. Besides, due to the adoption of an ontological domain definition, rules are modeled in a high semantic level that produces a good match between service and business process needs.

However, it has some challenges. First it requires service repositories be semantically structured, which is not easy to accomplish. Second, service composition is not addressed in this proposal, but it could be an incremental evolution.

Despite most discussions around context-aware applications are focused to mobile enviroments, our proposal use best parts of those to employ them in a business scenarios. Mobiles can be considered one element of business process, but to attend business needs we argue the necessity of a broader view. At that time, the study contributes to automate decisions when business process requires services to support business logic.

5 Conclusions and Future Work

The development of services is directly associated with business process. Service features (such as self-contained, low coupling, high cohesion, interoperability etc) provides flexibility and agility to business to attend market requirements. Besides, the use of service speed development, and decreases maintenance costs [2; 6]. As seen throughout this paper, business processes need the services to be aware of their context.

So, it was described an approach to perform the discovery of context-aware services, using AOP to ensure flexibility in capturing context of business. We showed an application scenario from which it is possible to conclude that the framework is flexible enough to identify (new) services aligned to business without code maintenance providing agility to time to market. In addition, the context improvements in services repository make suitable for semantic manipulation of its components.

As future work, we will perform a case study using benchmarking tools for web services available. We also intend to compare the contextual universe proposed by [3] with the results obtained in ongoing research.

References

1. Brézillon P., Pomerol J-Ch.: Contextual knowledge sharing and cooperation in intelligent assistant systems. In Le Travail Humain, v. 62, n. 3, (1999) 223 – 246.
2. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, Englewood Cliffs (2005)
3. Han, W., Shi, X., Chen, R.: Process-context aware matchmaking for web service composition. In Journal of Network and Computer Applications, v. 31, n. 4, (2008) 559-576.
4. Keidl, M., Kemper, A.: Towards context-aware adaptable web services. In Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, New York, NY, USA (2004).
5. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J., Irwin, J.: Aspect-oriented programming. In European Conference on Object-Oriented Programming, (1997) 220 – 242.
6. Josuttis, N. M.: SOA in Practice: The Art of Distributed System Design. O'Reilly (2007).
7. Laddad, R.: AspectJ in Action: Enterprise AOP with Spring Applications. Second edition, Greenwich, CT: Manning Publications Co (2009).
8. Marks, E. A., Bell, M.: Service Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology, Wiley (2006).
9. Martin, D.: Putting Web Services in Context. In Electronic Notes in Theoretical Computer Science, v. 146, n. 1, (2006) 3 – 16.
10. Papazoglou, M. P., Traverso, P., Dutsdar, S., Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenges. IEEE Computer Society, vol. 40, issue 11, (2007) 38 – 45
11. Prezerakos, G.N., Tselikas, N. D., Cortese, G.: Model-driven Composition of Context-aware Web Services Using ContextUML and Aspects. In IEEE International Conference on Web Services, ICWS, (2007) 320 – 329
12. Regev, G., Wegmann, A.: Business Process Flexibility: Weick's Organizational Theory to the Rescue. In Proceedings of the 7th BPMDS Workshop on Business Processes and Support Systems: Requirements for flexibility and the ways to achieve it, CAiSE Workshops (2006).
13. Sheng, Q. Z., Benatallah, B.: ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services. In Proceedings of the International Conference on Mobile Business, (2005) 206 – 212
14. Truong, H., Dustdar, S.: A survey on context-aware web service systems. In International Journal of Web Information Systems, v. 5, n. 1, (2009) 5 – 31
15. Xin, C.: Service-oriented architecture in business. In International Colloquium on Computing, Communication, Control, and Management, v. 4, (2009) 521 – 524