E-commerce Service Components with Multiple Classes of Service and Dynamic Adaptability Mechanisms

Vladimir Tosic, Bernard Pagurek

OCIECE-01-09
May 2001

E-commerce Service Components with Multiple Classes of Service and Dynamic Adaptability Mechanisms

Vladimir Tosic, Bernard Pagurek

Department of Systems and Computer Engineering, Carleton University
1125 Colonel By Drive, K1S 5B6, Ottawa, Canada
{vladimir,bernie}@sce.carleton.ca

Abstract. Dynamic engineering and adaptation of complex e- and m-commerce systems supports business agility and flexibility. This work-in-progress paper presents research on service components with several distinct classes of service, called service offerings, and their dynamic adaptation. Having multiple service offerings enables a service component to provide to every consumer an appropriate level of service and quality of service (QoS). It also supports different capabilities, rights, and needs of consumers of the service component. Service components and service offerings are described in a comprehensive XML-based specification that describes functionality, dependencies from other service components, functional and QoS constraints, authorization policies, as well as cost. Dynamic adaptation mechanisms that we explore are based on the manipulation of service offerings. Our mechanisms enhance robustness of the relationship between a service component and its consumer. This robustness is important in e- and m-commerce systems due to the issues of trust and customer retention.

1. Introduction

We define a service component as a composable, reusable, and replaceable self-contained unit of service provisioning and management. It encapsulates some business logic and data and can be reused in different service compositions. It can also be detached from the service composition and replaced with another appropriate service component, possibly from a different vendor. It is important to note that a service component can provide not only software functionality and data, but also access to some hardware resources like memory, printing, network bandwidth, etc. This software-hardware integration opens some issues that we are exploring in our research.

We make a distinction between a service component and a service – a service has some business-oriented meaning for an end user, while a service component has value primarily in service compositions and not when it is used in isolation. This distinction is context-dependent – the same unit of business logic and data can be a service component in one context and a service in another. We also make a distinction between a service component as a unit of service provisioning and implementation-level concepts like a software component and a distributed object. A service component is a higher-level concept and it can be implemented with one or many software components, with one or many distributed objects, with some combination of software com-

ponents and distributed objects, or with some other technologies, for example, mobile code. A service component can also be a composition of lower-level service components. Implementation of a service component is hidden from its consumers – consumers get information only about the provided services and eventually about the requirements--like the presence of other service components--that have to be satisfied in order to use this service component in a service composition.

Although service components can be also composed statically (i.e., during the design time), we are particularly interested in dynamic (i.e., runtime) composition of service components and dynamic adaptation of service compositions to various changes. Our goal is to better support flexibility and adaptability of complex systems like e-commerce and m-commerce software. Dynamic composition of service components enables more agile development of new services, with minimal interruption to running applications. On the other hand, dynamic adaptation of composite services is needed in order to accommodate changes that cannot be accommodated on lower system levels (like communication software, operating system, etc.). Such changes are especially significant in m-commerce and other mobile computing systems.

In this work-in-progress paper we present our ongoing research on service components with multiple classes of service and dynamic adaptation of their service compositions. This work is related to the work on dynamic service composition conducted in our research group [5]. We will illustrate our concepts on a gradually introduced e-commerce example. At the end of the paper we will summarize the relevance of our work for engineering of complex e- and m-commerce systems.

2. Service Offerings Representing Multiple Classes of Service

A service component can serve many different consumers, possibly at the same time. To improve flexibility and adaptability of compositions of service components, it can be useful for a service component to offer several different classes of service. These classes of service can differ in usage privileges, service priorities, response times guaranteed to consumers, verbosity of response information, etc. The concept of classes of service also supports different capabilities and rights of potential consumers of the service component. Further, different classes of service may imply different utilization of the underlying hardware and software resources and, consequently, have different prices. Multiple classes of service enable discrete customization of the received service and QoS and limit the complexity of required management. This approach does not exclude having customization of service and QoS by means of appropriate service parameters, but in the latter cases management can be more complex.

The issues of Quality of Service (QoS) and balancing of limited underlying resources are particularly motivating for having multiple classes of service. If the underlying resources were unlimited, all consumers would always get the highest possible QoS. Unfortunately, this is not the case, so is suitable to provide different QoS to different classes of service component's consumers. Service components that are provided by third parties on a pay-per-use basis are very illustrative in this respect. Their service component providers want to achieve maximal monetary gain with optimal utilization of resources. On the other hand, consumers want to receive service

and QoS they need and are willing to pay for, while minimizing the price/performance ratio. Consequently, providing different classes of service for a service component increases the chance of succeeding in the market because of the flexibility to accommodate several classes of consumer.

Let us now illustrate the previous discussion with an e-commerce example. Stock market notification can be implemented as a pay-per-use service component with multiple classes of service that differ in provided QoS (for example, the rate of notification or verbosity of provided information) and in price. On the other hand, some financial analysis software can also be provided as a pay-per-use service component with multiple classes of service to accommodate different types of its consumers. These two service components can be composed in a way that the stock market notification component provides information to the financial analysis component that in turn provides the results of its analyses (e.g., recommendations) to its consumers.

Our approach to providing multiple classes of service is based on the concept of service offerings. A service component can have multiple service offerings each representing a different class of service. Note that a service component can offer to its consumers one or more distinct interfaces. By "interface" we mean a unit of functionality (this term should not be confused with the homonymous concept in some programming languages). If a service component offers more than one interface, then service offerings are specified on two levels. First, for every interface one or more service offerings are defined. Then, service offerings of the service component are defined as allowed combinations of interface-level service offerings. Service offerings of one interface relate to the same functionality, but differ in authorization rights and QoS constraints, as well as cost.

We specify service components, including their interfaces and service offerings, in a comprehensive XML-based (eXtensible Markup Language) notation. We place a particular emphasis on the formal specification of different types of constraints – functional constraints (pre- and postconditions, and invariants), non-functional constraints (QoS guaranteed to consumers and QoS required from cooperating service components), and authorization policies (describing what subset of service component's functionality a service offering provides and under what conditions). Formal specification of functional constraints supports software reliability, reusability, maintainability, and evolvability. However, for service components we also need formal specification of other kinds of constraints [2]. We believe that as the number of service components on the market that offer similar functionality increases, the offered QoS and price/performance ratio, as well as adaptability, will become the main competitive advantages. Therefore, our comprehensive specification of service components supports choosing appropriate service components and service offerings, e.g., in the process of dynamic service composition. Additionally, it can also be beneficial to minimize unexpected feature interactions in service compositions. Note that, while a service offering contains specification of different types of constraints, these specifications are separated into multiple distinct layers in order to achieve greater flexibility and reusability.

3. Dynamic Adaptation Mechanisms

We also believe that dynamic adaptation mechanisms that are based on manipulation of classes of service, i.e., service offerings, are beneficial for both service component providers and consumers, especially in the case of pay-per-use relationships. Service component providers do not want to lose existing consumers when changes occur. If consumers have to find another service component to accommodate the change, they might chose one from a competing vendor. On the other hand, in many cases the change has to be accommodated quickly. In some cases, finding and choosing an appropriate alternative service component can turn out to be too slow and its success cannot always be guaranteed. Also, for some consumers it may be inconvenient to look for another service component every time the circumstances of operation change. Such a situation may occur in e- and m-commerce systems when choosing an alternative service component would require establishment of new trust relationships.

To avoid breaking the relationship between a service component and a consumer, we are researching dynamic adaptation mechanisms that are based on manipulation of service offerings and developing an appropriate management infrastructure and algorithms. We are exploring switching between service offerings, deactivation/reactivation of existing service offerings, and creation of new appropriate service offerings. The detailed discussion of these mechanisms and the corresponding infrastructure is beyond the scope of this work-in-progress paper, so we will only illustrate them on the previously introduced e-commerce example and note a couple of points.

In a turbulent stock market the adaptability of the relationship between the stock market notification component and the financial analysis component might be a very valuable feature. For example, depending on the analysis of the current situation, the financial analysis component could want to dynamically switch between different service offerings of the stock market notification component. Also, if a consumer of the financial analysis component wants to adjust the service it gets, this adjustment might require dynamic adaptation of the relationship between the two service components. If for some reason (e.g., mobility) there are some temporary disturbances of the communication between the two components, then the financial analysis component might have to temporarily deactivate some of its service offerings. These service offerings can eventually be reactivated after another change of circumstances. Dynamic evolution (versioning) of the stock market notification component can result in the need to dynamically create new service offerings for this service component, but also, as a result, for the financial analysis component. Note that creation of new service offerings is not creation of new functionality or new functional interfaces, but creation of new sets of QoS constraints and authorization policies for the existing functionality. However, this mechanism can be a useful support for dynamic evolution of service components, especially for describing effects on co-operating service components (the financial analysis component in the given example).

Also note that the issue with dynamic deactivation/reactivation of service offerings is what to do with the consumers using a deactivated service offering. We are developing support to automatically switch such consumers to an appropriate alternative service offering of the same service component and then notify these consumers about the change. A crucial aspect of this support is how to relate service offerings in order to decide on which service offering to switch. This issue can, but need not, be linked

to the issue of relating service offerings in order to enable their easier and more flexible specification. We are still exploring several possible alternatives for representing these relationships, including single inheritance, multiple inheritance, mixins, special tables, as well as various combinations of several mechanisms.

4. Conclusions and Future Work

Dynamic service composition and adaptation mechanisms support business agility and flexibility and therefore we believe that they can useful for some (if not many) e- and m-commerce systems. The concepts that we are working on support the flexibility and adaptability of e- and m-commerce systems in several ways. For example:

1. Multiple classes of service support different consumers with different characteristics, including power and/or type of devices they execute on. They also enable a service component to provide to every consumer an appropriate level of service and QoS and to better balance limited underlying resources.
2. For complex systems, like e- and m-commerce systems, manageability is a very important issue. As advocated in telecommunications service management, one advantage of having a relatively limited number of classes of service over other types of service customization is manageability.
3. Formal specification of various constraints supports more precise discovery of appropriate service components and classes of service.
4. Although the dynamic adaptation mechanisms that we are developing have limited power compared to finding alternative service components, they enable faster and simpler adaptation and enhance robustness of the relationship between a service component and its consumer. They enable a service component provider to retain existing consumers and also do not require establishment of new trust relationships between service components. We believe that these mechanisms are suitable for situations when the required adaptation is relatively limited and acceptable for the consumer. Examples of such adaptations are a small temporary degradation of service and a temporary switch to a more expensive class of service in order to sustain the overall level of service. These mechanisms are one way to accommodate some changes caused by mobile communications, which is an important issue in m-commerce systems.
5. Dynamic software evolution with minimal disruption to consumers is important for many e- and m-commerce systems. This is a very complex issue, but our mechanism and infrastructure for dynamic creation of new service offerings provide some support for describing effects of dynamic evolution on co-operating components. Another research project [3] within our research group is working on some other aspects of dynamic software evolution.

Apart from the work recently done in our research group on dynamic service composition [5] and dynamic software evolution [3], several other related works should be noted. First of all, some of our ideas, most notably providing different classes of service, are extrapolations and modifications of some telecommunications service engineering and management concepts. Particularly influential were the TINA (Telecommunications Information Networking Architecture) standard [4] and the work on

differentiated services called DiffServ [1]. While these works are specific to the telecommunications domain, our work tries to address issues of general service components in converged computing/communication systems. Another set of related work are several recent industrial initiatives, most notably Microsoft's .NET [6] and Sun's Open Net Environment (ONE) [9], that are based on the concept of a Web service. A Web service is a service component (in our definition) that communicates by means of XML-based standards. We concentrate our efforts on researching service-level issues that are currently not addressed by these industrial initiatives. Our work is not bound to service components that communicate using XML-based standards because we want to research issues that independent of the communication mechanism used. On the other hand, we find the work on adaptable software and dynamic software evolution, like the works reviewed in [7], to also be related to our work, although we explore dynamic adaptation mechanisms different from the architecture-based approaches. While our work tries to address some issues that are outside the scope of component-based software engineering, the work in this area, especially on supporting dynamism, like [8], has had an influence on our work. Our work on comprehensively describing service components is strongly influenced by the ideas from [2].

We have developed the main concepts of our approach, but there are still a number of important issues that we are currently working on. Most of these issues are related to the management infrastructure and algorithms supporting dynamic adaptation in various situations, including dynamic evolution of service components. Examples are the infrastructure support for minimizing the overhead of dynamic switching, mechanisms for relating service offerings in order to support their easier specification and automatic switching, and the infrastructure for dynamic creation of new service offerings. We are working on a prototype implementation of our concepts.

References

1. Aimoto, T., Miyake, S: Overview of DiffServ Technology: Its Mechanisms and Implementation. IEICE Trans. Inf. & Syst., Vol. E83-D, No. 5 (May 2000) 957-964
2. Beugnard, A., Jezequel, J.-M., Plouzeau, N., Watkins, D.: Making Components Contract Aware. Computer, IEEE Computer Society Press (July 1999) 38-45
3. Feng, N., Ao, G., White, T., Pagurek, B.: Dynamic Evolution of Network Management Software by Software Hot-Swapping. Accepted at IM 2001 (Seattle, USA, May 2001)
4. Kristiansen, L. (ed.): Service Architecture, Version 5.0. TINA-C (June 16, 1997)
5. Mennie, D., Pagurek, B.: An Architecture to Support Dynamic Composition of Service Components. Presented at WCOP 2000 (Sophia Antipolis, France, June 2000). On-line at: http://www.ipd.hk-r.se/bosch/WCOP2000/submissions/mennie.pdf
6. Microsoft Corporation: Microsoft .NET. WWW page (2001). On-line at: http://www.microsoft.com/net/
7. Oreizy, P., Medvidovic, N., Taylor, R. N.: Architecture-Based Software Runtime Evolution. In Proc. of ICSE'98, ACM Press (Kyoto, Japan, April 1998) 177-186
8. Pryce, N., Dulay, N.: Dynamic Architectures and Architectural Styles for Distributed Programs. In Proc. of FTDCS'99, IEEE Computer Society Press (Cape Town, South Africa, December 1999)
9. Sun Microsystems: Sun Open Net Environment (Sun ONE). WWW page (2001). On-line at: http://www.sun.com/software/sunone/