Motivation

Service-Oriented Architecture (SOA) is an architectural style that emerged as the answer to the latest requirements for loosely-coupled distributed computing. Inline with the conventional distributed computing paradigm, functionality is decomposed into distinct architectural elements, distributed over the network. Nevertheless, in SOA the basic architectural elements (a.k.a services) are by themselves autonomous systems that have been developed independently from each other. We work on two areas in this domain :

Supporting Maintainability in Service-oriented Software. Until now, state of the art research in SOA systems has focused mostly on issues concerning the construction phase of service-oriented software. The outcome of these research efforts was mechanisms for discovering, composing and accessing available services. However, several other phases of the development process are currently underdeveloped. Specifically, we concentrate on the maintainability quality attribute. The importance of this issue is evident towards the success of the SOA paradigm, which promotes the development of software consisting of independently evolving basic engineering elements that may further vary in quality (e.g., performance, availability, reliability).

Reducing the Complexity of Service Substitution. Service substitution is a key issue towards dealing with the independent evolution of services along with their variation in quality (e.g., performance, availability, reliability). Research efforts that focus on service substitution can be divided in two categories: (i) abstraction-based and (ii) adapter-based approaches. While considering adapter-based approaches the following issue is raised: the effort and time required by the service substitution process scales up with the number of available services that should be examined as potential candidate substitutes of the target service. This problem is a serious drawback towards a practical service substitution approach if we consider that the service substitution process involves human intervention to validate the mapping between target and substitute services.

Research

Supporting Maintainability in Service-oriented Software. In conventional Object-Oriented (OO) software, maintainability can be improved by employing well known fundamental design principles such as OCP (Open Closed Principle), DIP (Dependency Inversion Principle) and LSP (Liskov Substitution Principle).We revisited these principles in the context of the SOA paradigm and argue about the need to adapt/refine them to the specificities that characterize the paradigm. Specifically, we:

- examined the maintenance scenarios that can be handled by the conventional use of the fundamental design principles in the SOA paradigm and discuss why these scenarios are not realistic,

- adapted/refined the fundamental design principles such that their use in service-oriented software becomes effective towards handling realistic maintenance scenarios, and
- sketched the ForeverSOA infrastructure, which aims at facilitating the adoption of the refined principles in the development of SOA software. The prominent concept in ForeverSOA is a reverse engineering process that recovers service abstractions out of available services. An abstraction characterizes a group of services, providing similar functionalities via different interfaces and serves for developing software that may access any of the grouped services without depending on their interfaces.
-

Reducing the Complexity of Service Substitution. Our work shares the objective of adapter-based approaches. However, our specific goal is to reduce the effort and time required to achieve this objective. To this end, we propose a hybrid approach [16] that borrows ideas from abstraction-based approaches so as to handle the complexity of service substitution. The proposed approach relies on a formal foundation that comprises two substitution relations and corresponding substitution
theorems, which are in line with the Liskov substitution principle (LSP). Based on the proposed relations and
theorems, available services are organized into groups, characterized by abstractions, called profiles. Then, the complexity of service substitution scales up with the number of available profiles, instead of scaling up with the
number of available services. Our experimental results highlight the aforementioned benefit. Currently, we are working towards a reverse engineering process that would allow to improve the organization of services into groups, by recovering service abstractions from a set of available services. The proposed framework may be
extended to account for mismatches in the order of operations; it may also be combined with keywords-based and QoS-based search techniques

Contributors

- Dionysis Athanasopoulos
- Apostolos Zarras
- Valérie Issarny

Supporting Grants

- ForeverSOA associated team

Software Download

- To come soon

Presentations (registered users only)

Publications

Titre Service Substitution Revisited AuteursAthanasopoulos Dionysis; Zarras Apostolos V.;

Issarny Valérie DétailIn *24th IEEE/ACM International Conference on Automated Software Engineering - ASE 2009* (16/11/2009) Accès au texte intégral

Titre CoWSAMI: Interface-aware context gathering in ambient intelligence environments
AuteursAthanasopoulos Dionysis; Zarras Apostolos; Issarny Valérie; Pitoura Evaggelia; Vassiliadis Panos Détail
Pervasive and Mobile Computing
4,3 (2007) 360-389 Accès au texte intégral