Mixing Workflows and Components to Support Evolving Services

Françoise Baude, Université de Nice Sophia-Antipolis, France
Virginie Legrand, Université de Nice Sophia-Antipolis, France
Ludovic Henrio, Université de Nice Sophia-Antipolis, France
Paul Naoumenko, Université de Nice Sophia-Antipolis, France
Heiko Pfeffer, Technische Universität Berlin, Germany
Louay Bassbouss, Technische Universität Berlin, Germany
David Linner, Technische Universität Berlin, Germany

ABSTRACT

Composite distributed services involve local and remote services that get orchestrated according to specific business logic. This logic can be programmed by applying a traditional general-purpose programming language, but is generally described using a workflow language that coordinates a set of given services. The services involved in the composition, or the composition may need to evolve both at the business logic level (workflow level) and the global architecture level. This paper presents a solution to ease such evolution for compound distributed services and the authors’ proposal enables the evolution of both the business logic and the underlying architecture. This paper suggests relying on a distributed software component model to represent and easily manage the set of local or remote software entities (services) involved in the composition. Composite services are represented in a model that combines the use of a distributed and hierarchical software component model and new timed-automata based workflow language. This combination makes explicit the separation between functional and non-functional concerns, and as a consequence this approach helps in defining the required and various evolution procedures in context to compound services.

Keywords: Components, GCM, Service Composition, Service Evolution, Workflows

INTRODUCTION

General Context

Composing services and allowing services and their composition to evolve at runtime is crucial for most of distributed applications. Indeed, as soon as the execution environment may change in an unplanned manner, services must be able to adapt to those changes. In a more challenging manner, one could also expect services to evolve automatically in order to provide better quality of services, matching new user requirements, or providing users with unforeseen functionalities.

DOI: 10.4018/jaras.2010100105
The context of the research work presented in this paper is the BIONETS European project, aiming at bio-inspired evolution of services and network protocols. In this context, we came out with a new composition model, integrating both workflow-oriented orchestration and component-oriented composition. Additionally, our composition model features high evolution and adaptation capacities. In this paper, we present this new model, the architecture it features, and the evolution capabilities it enables.

The originality and value-added of our model comes from the mixing -- and thus implied respective advantages -- of two service composition models, each specially focusing on some aspects of service composition: the workflow based composition approach mainly manipulates timed sequences of service invocations, whereas the component based approach focuses on the dependencies between ports of the different services forming a compound service.

A consequence and a particularity of this mixed composition model is that, as each of the model focuses on evolution, the compound model benefits from their respective evolution capabilities. Consequently, our model is particularly targeted at evolution of services and of their composition.

**Viewpoint and Motivations**

**Components or Services**

A first crucial aspect of the positioning of our work is the use of a component model as the service composition paradigm. The reasons for this choice are that components can easily encapsulate services, and that component-based architectures are better structured, so easier to adapt than pure service-based service oriented architectures. This point of view is already adopted by SCA where the needed services are seen as components within the resulting Service-Oriented Architectures (Beisiegel et al., 2005). Compared to more traditional ways of composing services together, a component encapsulating a service makes explicit its dependencies towards other services (components). Consequently, a component composition model permits a complete and very structured view of the complete application’s architecture that involves services.

**Components and Workflows**

A component can be implemented and then published and accessed using different technologies and protocols. It can for instance be written in Java, and accessible through the Java RMI protocol; it can be written in a script-like or XML-based language, as WS-BPEL, and accessible through the use of Web Services or REST protocols, etc. So, this versatility regarding the used technology is a plus, in that, in a Service Oriented Architecture built using components, it will be easy to mix heterogeneous components regarding their supporting technologies. In this paper, we specifically concentrate on the case where a service underlying logic can be specified as a flow of activities expressed in a workflow language (e.g., WS-BPEL). At runtime, one or several workflow engines should be associated to the execution of those flows. To take full advantage of the evolution capabilities gained by the reliance on a component-based approach, we also consider the workflow engine and every dependency to a service as components in the resulting executable architecture.

Thus, from an execution and service specification outlook, components form the framework to assemble services along a spatial viewpoint, whereas workflows are convenient to define individual services and to coordinate different service invocations, along a rather temporal viewpoint. This clear separation of roles also corresponds to the view adopted by SCA-BPEL (Beisiegel et al., 2005). In other words, we champion the idea that component-based composition overlays service composition expressed by the means of orchestration languages.

**A Two-Level Evolution**

The main novelty of this paper is to promote the mixing of workflow and component paradigms in order to fully enable evolution of complex applications. Evolution can thus occur at each
Related Content

A Heuristic Approach for Multi Objective Distribution Feeder Reconfiguration: Using Fuzzy Sets in Normalization of Objective Functions
www.igi-global.com/chapter/heuristic-approach-multi-objective-distribution/66818?camid=4v1a

Principles of Classification
www.igi-global.com/chapter/principles-classification/69672?camid=4v1a

Evolutionary Growth and Control in User Tailorable Systems
www.igi-global.com/chapter/evolutionary-growth-control-user-tailorable/4213?camid=4v1a
Efficient Adaptation Decision Making Algorithms for Context-Aware Applications
www.igi-global.com/article/efficient-adaptation-decision-making-algorithms/46033?camid=4v1a