To offer a comprehensive response to the above concerns, researchers in the VTT Software Architectures and Platforms department are developing a VantagePoint tool, which will make the contextual semantic information related to service descriptions easier to understand, and its use by an application developer foolproof. The tool allows users to view ontology instances associated with complex contextual information in a more illustrative and comprehensible way. It also allows users to semantically model and interactively simulate contextual environments of interest. These may be either physical real-world (ie devices, services, functional capabilities of service, contexts) or conceptual (business boundaries, networking or security domains). It supports the conceptual design of applications (eg verifying a service composition logic) or middleware-level services (eg semantic service discovery) against one or more contextual scenarios. Moreover, we believe that the research presented is a step toward the better understanding and wider acceptance of ontology-based semantic technology also for non-Web services.

VantagePoint is written in Java; it uses the Jena interface to manage OWL ontologies, and Java 2D graphics to visualize them. VantagePoint can have several visualization libraries containing domain-specific icons. These libraries are stored as simple text files that contain URLs of the icon files providing isometric visualization from different perspectives (PNG images), and a URL of a semantic class description in one of VantagePoint's semantic libraries. A browser tool is provided for examining the visualization libraries. While the current libraries relate to intelligent home applications, future elements will describe other intelligent environments such as car, plane, or mobile outdoor domain.

Links:
http://www.hitech-projects.com/euprojects/amigo/
http://www.vtt.fi/proj/vantagepoint/

Please contact:
Julia Kantorovitch, Jarmo Kalaoja
VTT, Finland
Tel: +358 20 722 2334
E-mail: Julia.Kantorovitch@vtt.fi, Jarmo.Kalaoja@vtt.fi

Diapason: An Engineering Environment for Designing, Implementing and Evolving Service Orchestrations

by Frédéric Pourraz and Hervé Verjus

The aim of Diapason is to allow designers of service-oriented architectures (SOAs) to precisely orchestrate services using an orchestration language based on pi calculus. Once defined, an orchestration can be verified against properties and constraints, can be deployed as a new service and can evolve dynamically and on the fly.

SOAs are service-based applications for which classical software engineering approaches fail. This is due to the heterogeneous, autonomous, widely distributed and loosely coupled nature of the services. In other words, when building an SOA, the designer does not control the service's implementations. As SOAs are increasingly used to support widely distributed software-intensive systems in a plethora of domains (business, manufacturing, health, Grid-based applications, military etc), the design, implementation and evolution of SOAs is a challenging problem.

Diapason Approach

We introduce a new environment called Diapason. Diapason is an SOA-based systems-engineering environment that supports service orchestration, along with the analysis, deployment, execution and evolution of that orchestration. Diapason provides a layered formal language called pi-Diapason, which relates to SOA structure/topology with most of the service orchestration patterns already proposed (www.workflowpatterns.com). It is also extensible (ie architects can define their own extensions and concepts) and executable.

It also provides:
- a properties definition language called logic-Diapason
- an analyser (a properties checking tool)
- an animator (a graphical simulation tool)
- deployment mechanisms
- a virtual machine (a runtime engine).

Basically, services provide operations that can be remotely invoked. The manner in which the services (and operations) are implemented is unimportant, since services are considered to be black boxes that can be composed (orchestrated). When orchestrating Web services, WSDL files are referenced in order to obtain the signatures of the operations and to then invoke these operations at runtime.

Once defined, a service orchestration can be deployed as a new service that can be further reused in other orchestrations and/or can be modified according to new requirements. When an orchestration is reused it is considered to be a standalone service, providing its own operation(s). Hence, the orchestration can be composed with other services and be involved in other orchestration(s). A service orchestration is executed by way of a virtual machine that interprets pi-Diapason language. In this way, pi-Diapason can be used as a formal and executable SOA specifications language.

Service composition is a key issue in Diapason. Pi-Diapason is formally based on polyadic high-order Milner's pi-calculus, which is a form of process algebra. Service orchestration descrip-

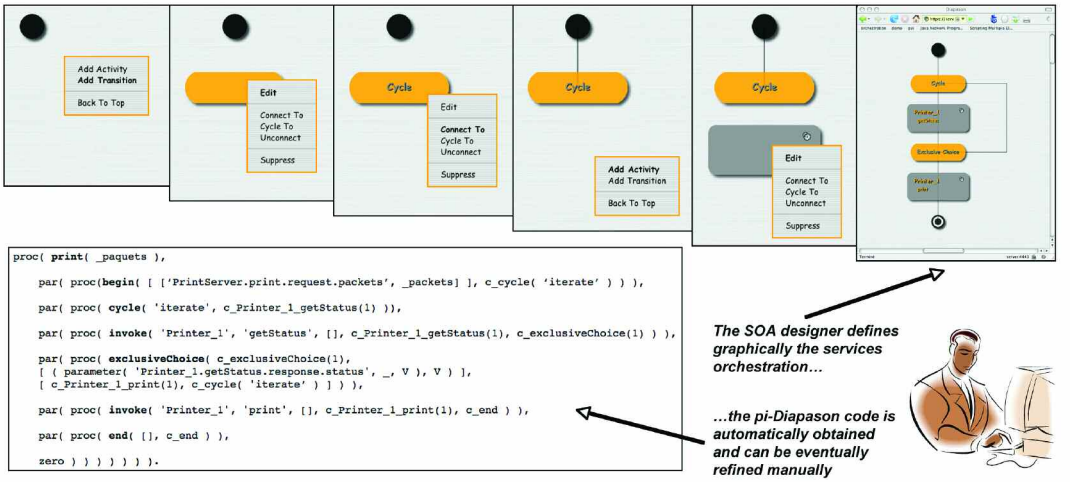

Figure 1: Orchestration description (Diapason graphical editor).

tions use orchestration patterns that are formally defined in terms of pi-calculus processes. Basically, a service orchestration is a complex pi-calculus process that composes other pi-calculus processes. In order to simplify the description of an orchestration, Diapason provides a lightweight and intuitive graphical editor.

Thanks to the pi-calculus mobility (introduced in the first-order pi-calculus but extended to behaviour mobility support in the high-order pi-calculus), we can dynamically modify the service orchestration at runtime without stopping the execution of this orchestration. Evolving a service orchestration at runtime is also very challenging. Evolving a Diapason service orchestration is the same as evolving a pi-calculus process.

Pi-Diapason provides pi-calculus-specific constructs that are responsible for evolution: using these constructs, the SOA architect can interact with the pi-Diapason virtual machine in order to modify the service orchestration pi-Diapason code at runtime. Modifications are transmitted to the virtual machine and are then dynamically applied without interrupting the executing orchestration (the virtual machine supports services orchestration state consistency management). The service orchestration behaviour is changed on the fly by way of pi-calculus messages that contain the required changes.

SOAs defined using pi-Diapason can be checked against properties (eg deadlock-free, vivacity, liveness, structural and behavioural properties). Properties

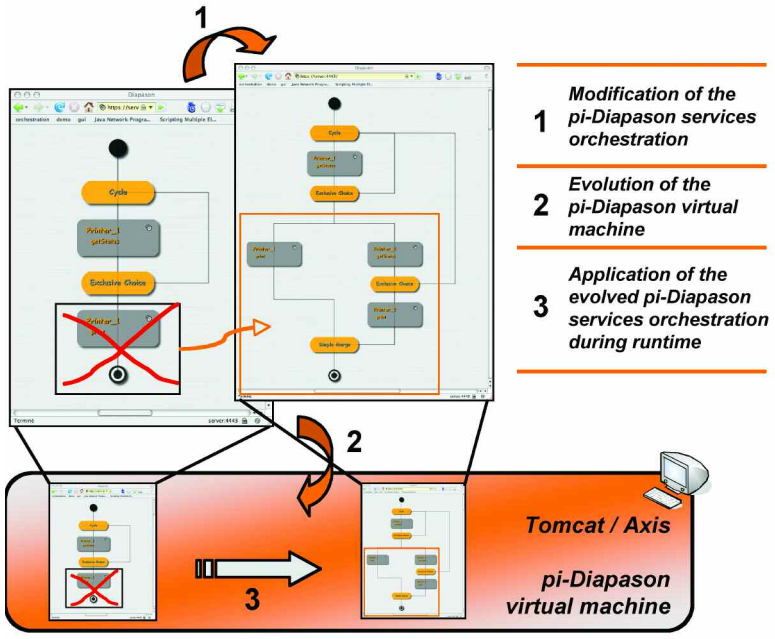

Figure 2: Orchestration evolution.

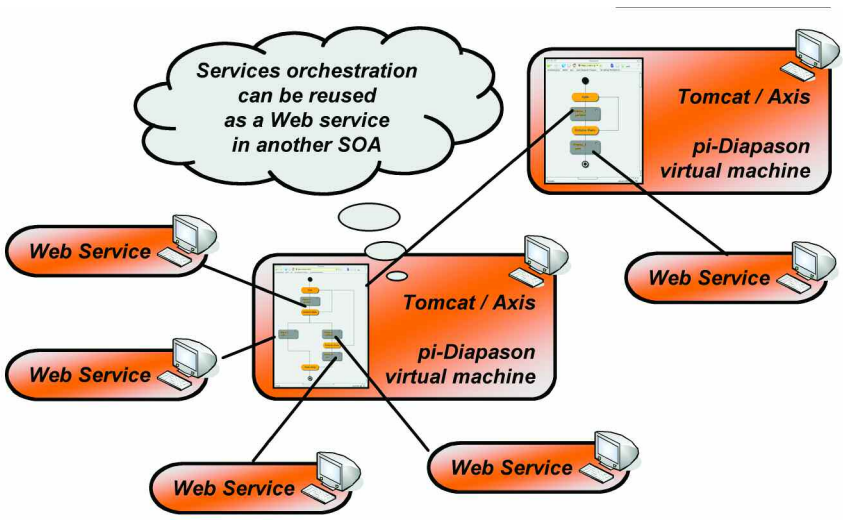

Figure 3: Orchestration reuse.

are defined using logic-Diapason, a logic-based properties definition language. SOA properties are then verified according to two steps: the first is performed by the pi-Diapason virtual machine that allows the extraction of all possible execution traces, while the second is done by the analyser, which allows properties expressed using logic-Diapason to be checked over all the previously extracted traces. In addition to this formal verification, the animator lets us simulate one or more of the SOA's execution traces in a graphical manner. This is a more intuitive (though informal) way of checking the SOA's structure and behaviour.

Diapason Engineering Environment

A Diapason service orchestration is deployed as a service; this embeds the service-orchestration pi-Diapason description, a pi-Diapason virtual machine, in order to interpret the orchestration and a context-aware deployment platform (according to the application server on which the service is deployed, ie Tomcat Axis or similar).

Please contact:
Frédéric Pourraz and Hervé Verjus
University of Savoie - Polytech'Savoie, LISTIC, France
E-mail: frederic.pourraz@univ-savoie.fr, herve.verjus@univ-savoie.fr

SUPER – Raising Business Process Management back to the Business Level

by Christian Brelage, Ingo Weber and Alina Dima

The SUPER project (Semantics Utilized for Process Management within and between Enterprises) attempts to make a quantum leap in business process management by improving modelling and management of business processes. This leap will be achieved by integrating and utilizing semantic technologies for business process management. It answers the two most urgent issues emerging from BPM: shifting control of processes from IT professionals to business experts and carrying up business process management to a new complexity level.

SUPER is a European Union-funded Integrated Project (IP) with a duration of 36 months and under the coordination of SAP. It started in April 2006 and unites 19 partners and approximately 60 researchers. The project consortium is a balanced blend of technological industry partners, service providers and academic research teams from all over Europe.

The motivation behind SUPER arises from the challenges the increased frequency with which business models and the contexts of enterprises change in today's world. The causes for these changes are mostly new internal or external business requirements – such as closer integration with suppliers and customers, implementation of new standards, deployment of new application components – which may originate in emerging business opportunities or new regulations from legal bodies. These challenges induce two requirements: to provide fast and cheap access to the space of processes in an organization, and to enable swift adaptation of operational business processes.

The major objective of SUPER is to raise Business Process Management (BPM) to the business level where it logically belongs, from the IT level where it mostly resides now. This objective requires that BPM is accessible at the knowledge level of business experts and business analysts. Semantic Web and, in particular, Semantic Web Services (SWS) technology offer the promise of integrating applications at the semantic level. Therefore, this project aims at providing a semantics-based and context-aware framework, based on

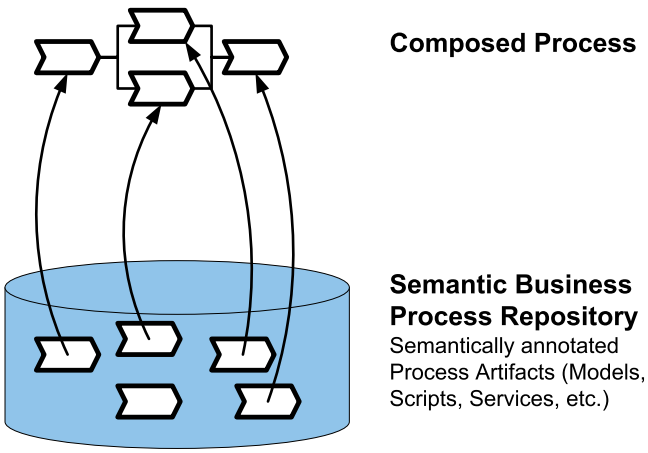

Figure 1: Business Process Composition based on semantic annotations of services and processes.

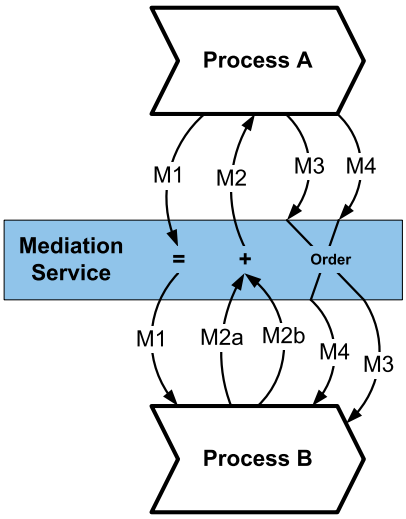

Figure 2: Business Process Mediation deals with heterogeneity in the behavioral interfaces and message formats of processes.