SPACEFLIGHT - A VERSATILE LIVE DEMONSTRATOR AND TEACHING SYSTEM FOR ADVANCED SERVICE-ORIENTED TECHNOLOGIES

Josef Spillner
Technische Universität Dresden
Faculty of Computer Science, Chair for Computer Networks
01062 Dresden, Germany
E-mail: josef.spillner@tu-dresden.de

Abstract – New research advances in loosely-coupled, uniformly described service-oriented systems continue to appear at a high rate. Application areas such as enterprise information systems, cloud computing and programmable web applications benefit from these developments. Yet, it is hard for researchers, developers and students to understand the combined effect of methodologies, which leads to unutilised opportunities to further accelerate the innovation. We present our proposed solution called SPACEflight. It is a free, ready-to-use demonstrator which contains a complete service platform, service engineering tools, extensions for cloud computing and service quality ensurance, and on top a user-friendly service-oriented desktop with tradeable scenario services.

I. Introduction

Service orientation is an established paradigm in the design of programmable application software integration and beyond, even applied to workflow systems, collaborative mobile agents and cloud computing stacks. The fundamental advantages of service-oriented architectures encompass flexible and dynamic composition, substitution and evolution of constituent parts of the overall distributed system. The fundamental techniques include uniform and formal descriptions, both for static interfaces and dynamic protocols, as well as transparent directories with search and matchmaking capabilities based on the descriptions.

In recent years, numerous advanced service orientation concepts have been developed. For instance, the service execution can now be protected by negotiable contracts and adapted in the event or providence of contract violation. The invocation can be triggered by ad-hoc generated user interfaces and custom service frontends registered on open service marketplaces. Furthermore, a plethora of new description formats beyond just service interface declarations is backing these advances - USDL, WADL, WSML, WSAG and other acronyms are regularly announced by scientists all over the world.

Unfortunately, most of the advances remain abstract ideas on paper. There are few systematic approaches to bundle innovative ideas from service orientation concepts and see how well they work in a combined manner beyond teaching foundries [1]. Students and software developers are confronted with finding this out by themselves over and over again, with practical issues such as unavailable implementations for many concepts.

We propose to counter this problem with SPACEflight, a free, ready-to-use demonstrator for advanced service-oriented technologies aimed at researchers, developers and students.

II. Demonstrator Design and Architecture

SPACEflight is a stand-alone software demonstrator based on a universal operating system (Debian GNU/Linux) which provides all the required low-level system facilities for controlling the hardware including attached displays. Its bootloader is configured to run the system in either full graphical demonstration mode or special-purpose server modes restricted to service management and/or execution functionality. The latter can be used to run several execution instances simultaneously and connect them to a central management instance, thus gaining a distributed service execution platform for experiments.

As central eponymous software package collection, the demonstrator contains SPACE, the modular and extensible Service Platform Architecture for Contracting and Execution [2]. A packaging layer configures and integrates SPACE parts into the operating system. The system's startup scripts are modified in a way that all SPACE platform services are started through virtual detached daemon sessions which can be reattached interactively by the demonstrator user at any later time.

In the case of a full demonstration, the last-run startup script starts the desktop environment (KDE). The desktop has been customised with a suitable wallpaper, a set of folders divided by roles and icons to all interesting platform services, web interfaces, log files and documents within the folders. The major roles of the demonstrator user are service platform operator, service provider and service consumer. Other roles such as service engineer are subsumed implicitly. All roles can be adopted by the demonstrator user in parallel.

The service lifecycle [3] encompasses the creation of services with both implemented executable and modelled interpretable artefacts bundled in a package, their extraction and deployment on the service platform by the provider, their selection and contracting by the consumer and the generation of usage data by both the system and the consumer. Figure 1 shows a typical service lifecycle, starting from service engineering tools over provisioning to actual usage. The diagram omits the obvious manual feedback loop back to the service engineer as well as automatic feedback loops such as adapting non-functional property specifications in description documents.

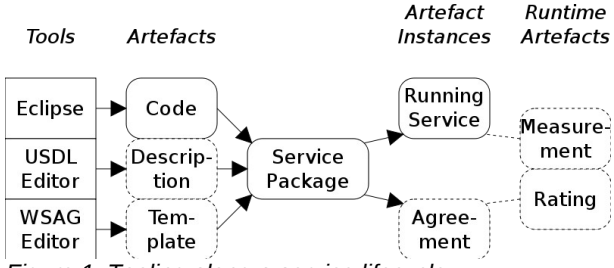

Figure 1: Tooling along a service lifecycle

A number of prepared scenario service packages are available to enter the service lifecyle in its runtime phases by skipping the modelling, development and engineering of services. The packages include both the implementation and the description artefacts, among them a service frontend descriptor, user interface generation hints and a graphical symbol.

The chosen design leads to the following orthogonal divisions of concerns within the demonstration system:

- Service lifecycle phase: engineering, offering, consumption, usage, feedback
- Platform service: central facilities - service registry and discovery, service deployment and lifecycle management, contract management, rating; decentral facilities - access control, execution, monitoring and adaptation
- Service tier: software service, database, web or desktop interface, associated log and configuration files

- Role: service platform operator, service provider or consumer

Each demonstration then consists of a trail through these aspects at any suitable level of detail in each step. This flexibility allows the user to concentrate and deep-dive on aspects of interest while still being able to follow the context around the chosen focus.

A number of SPACE platform extensions and auxiliary additions are integrated into the SPACEflight system distribution. They underline the aim to keep the core platform functionality to a minimum while offering sufficient extension points. The following extension sets exist:

- Internet of Services desktop: On top of the demonstrator scenario desktop, the consumer processes to search for services and negotiate contracts have been realised as desktop-integrated dialogues. Furthermore, the contract negotiation initiates a selection of service frontends through an integrated delivery concept. This covers stand-alone clients, desktop widgets and generated user interfaces.
- Cloud computing: Virtual machines are treated like very large service packages which can ship their own descriptions. The service deployment handler integrates with the deployment of a cloud platform (Eucalyptus), whereas the contract negotiation integrates with an instantiation of the virtual machine in the cloud, making use of a custom profile derived from the SLA. This extension increases the minimum required memory from about 2 GB to about 8 GB of RAM.
- Service engineering: Modelling and development tools (Eclipse, USDL and WSAG editors, and others) are made available to the service provider.
- Service quality: A further platform service called Metadata Correlation Service retrieves, aggregates and correlates information from the platform's query interfaces, such as the registry and the monitoring database. A quality metric is determined by finding discrepancies among non-functional property descriptions. The results are propagated back as quality attributes.
- Experimental facility: Crawler scripts populate the service registry with existing service descriptions found on the Internet. Through user activity simulators and measurement scripts, a number of experiments can be performed directly on a running SPACEflight instance. This extension also opens up the network of the otherwise fully protected demonstrator.

Additional extension points within the SPACE platform are defined by the heterogeneity of service package and artefact formats. This makes it rather trivial to add deployment support for new package types or interface descriptions.

The resulting architecture is shown in figure 2.

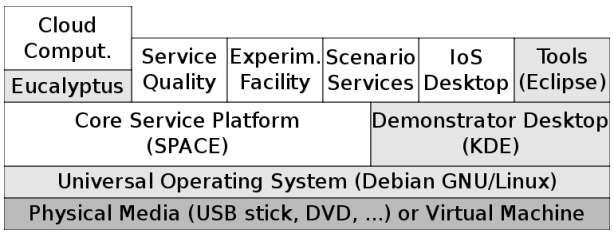

Figure 2: Composition of platform services, extensions and tools to SPACEflight

III. Teaching Potential

The demonstrator can be used for various purposes by combining the integrated SPACE extensions and additions appropriately. It has already been used to prepare large-scale service artefact assessment experiments and as ready-to-use part of another demonstrator in the area of cloud computing. A recent addition is the use within lectures and practical exercises so that students can experience advanced service concepts.

In addition to the engineering and provisioning tools extension, documentation for the platform services and regarding the service development process is included. Tutorials and videos turn the demonstrator into a self-learning environment. As the platform services are largely implemented in interpreted scripting languages, students even get the chance to inspect and instantly modify the platform code to understand why and how certain effects occur when they trigger them. The integrated revision control system (git) makes it easy to control deviations from the original code on a running system. Besides, at any time the original overall system state can be regained by a reboot due to the use of an in-memory overlay file system (aufs).

By running a social product network which is connected to the SPACE platform services, the resulting social service network allows the students to act as providers and consumers of services [4]. This way, they can cross-test their services with clients of other students and vice-versa. Also, network effects such as price specifications, self-marketing and technology domination can be experienced playfully by the students.

In addition to the social network portal, an automated static service quality checker can be integrated to identify incomplete or erroneous submissions during the offering phase. Tasks and collaboration features in a manual checker based on the ReviewBoard software allow students to track their practical exercise progress and comment and improve each others' solutions. Integration with a distributed revision control system with hooks for service package creation out of individual artefacts even combines learning experience from software and service engineering.

IV. Discussions

After about two years of development, SPACEflight has matured from a custom hand-crafted aggregation of various service platform components to an automatically-built live demonstrator which ships a number of preconfigured and integrated tools around a capable service platform. The build scripts ensure a repeatable high quality by eliminating regressions due to human mistakes even when adding further components. The quality and openness is being rewarded by ongoing rapid prototype development projects based on the service platform. SPACEflight has now been demonstrated successfully at 15 scientific conferences and exhibitions. With the latest additions, students can better understand the intrinsic values and problems of service lifecycles. The demonstrator's increasing use in higher education will mark the next milestone towards open and pervasive services distributed through global service platforms.

V. References

[1] *Paik H-Y., Rabhi F. A., Benatallah B., Davis J.* Service Learning and Teaching Foundry: A Virtual SOA/BPM Learning and Teaching Community. 8th International Conference on Business Process Management, September 2010, USA.

[2] SPACE Website: http://serviceplatform.org

[3] *Ferreira D.* R. Business Networking with Web Services: supporting the full life cycle of business collaborations. Handbook of Enterprise Systems Architecture in Practice, pp. 419-433, Information Science Reference, March 2007.

[4] *Spillner J., Caceres A., Buder B., Kursawe R., Globa L.S., Schill A.* Extending Social Networks with Service Delivery Capabilities for User-Centric Service Trading. 10th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science - TCSET, Lviv-Slavske, Ukraine, February 23-27, 2010.