ON THE USE OF WEB SERVICE COMPOSITIONS FOR SYSTEMS MANAGEMENT AUTOMATION: A USE CASE

Dimka Karastoyanova
Technische Universität Darmstadt, Computer Science Department
Hochschulstrasse 10, 64289 Darmstadt, Germany
dimka@gkec.tu-darmstadt.de

Alejandro Buchmann
Technische Universität Darmstadt, Computer Science Department
Hochschulstrasse 10, 64289 Darmstadt, Germany
buchmann@informatik.tu-darmstadt.de

ABSTRACT

In this paper we address the issue of automating systems management. Currently, any changes made in the configuration settings of the IT infrastructure in an enterprise are performed manually. To enable automatic system reconfiguration we propose the use of Web Service-enabled enterprise registries in combination with Web Service-based processes. Web Service-based processes define complex sequences of tasks and thus could implement complex reconfiguration rules. We call these composite Web services "systems management rules". Creating systems management rules can be automated using templates.

KEYWORDS

Systems zero-maintenance, automation, registry, Web Services, WS-flows, systems management rules

1. INTRODUCTION

Nowadays, systems (re)configuration and management still relies only on the capabilities of a system administrator. One of the reasons is the considerably greater importance of creating software systems and integrating them, than caring about automating their configuration. Due to the mix of IT infrastructure entities an enterprise owns a great effort is put in integrating them. However, apart from providing interoperability, integration involves a serious reconfiguration work.

Systems configuration and management is a matter of global control over all systems in an enterprise. We propose the use of a central registry for recoding the enterprise configuration settings, rules for reconfiguration, dependencies etc. A registry has to communicate any changes in configuration data to all affected systems and thus cause their reconfiguration. To provide for seamless communication between registry and systems we advocate the use of a configuration registry equipped with Web Service (WS) capabilities (section 2). The sequence of steps necessary to enforce reconfiguration rules could be either simple and passive, or much more complicated and require a more active role on the part of the registry. In the context of WS-enabled enterprise registries process-based composite WSs are the most appropriate technology to apply (section 3). This technology combines the advantages of traditional workflows and WSs. Open issues and conclusions are also presented (sections 3 and 4).

2. USING WSs TO FACILITATE SYSTEMS CONFIGURATION

Enterprise registries appeared as a result of the advances in several technologies, e.g. various directory service technologies, middleware registries, and registries holding users authentication data and rights. An

enterprise registry can be used for automated systems management, i.e. *zero-maintenance*, for the following reasons: registries have *global scope*; registries are capable of storing configuration *data* and *rules* in a centralized and consistent manner; registries can be equipped with functionality for *enforcing configuration rules* on all enterprise systems (thus foster automatic rules enforcement); a registry can *communicate* with all enterprise systems to reconfigure them [Jablonski, Petrov, 2004].

Based on configuration data and their relationships system administrators define configuration *rules* and enforce those rules manually on the enterprise IT infrastructure by performing sequences of reconfiguration tasks. Automatic enforcement of rules can be implemented by the registry in terms of on-purpose built-in functionalities including: detecting dependencies and resolving conflicts in configuration parameters across systems. This functionality is meant to keep the configuration data in the registry consistent, i.e. registry coverage. In order to apply rules directly on systems, without human intervention, the registry and the systems have to interact with each other, i.e. the results of the work done to maintain consistency have to spread over the affected systems. This communication is hampered by the heterogeneous nature of the environment – a typical integration problem previously addressed by computing paradigms such as Middleware, EAI, and B2B. The newest approach is the WS technology [Alonso et al., 2003]. We believe it is possible to simplify and facilitate the communication between registry and systems using this technology.

Consider an example of a simple infrastructure for Web applications (Figure 1, case 1): application server, a database, HTTP server, a firewall, and a registry storing configuration data (e.g. firewall settings).

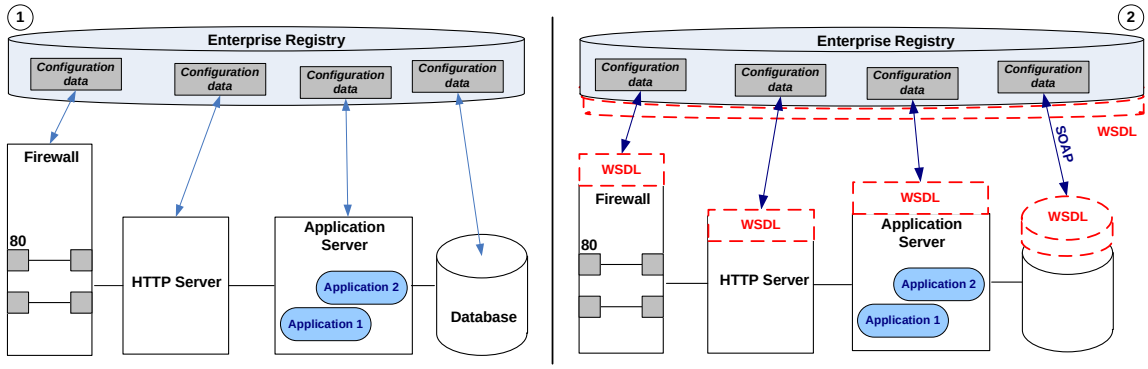

Figure 1. Applying WSs for seamless communication[1]

The firewall is configured to allow access to the applications on the application server only through port 80; that information is represented at the registry. All other components of the infrastructure can query the registry and find data about themselves and about all other components. The registry has to communicate to all systems to reconfigure them. This is only possible via multiple on-purpose adapters [Alonso et al., 2003] that perform the mapping between different protocols and data exchange formats. Developing the adapters is a hard work and consumes investment in terms of time for development and resources.

The number of protocols and formats that a registry would have to support can be reduced by simply utilizing the advantages of the open, XML-based Web Service standards [Bellwood et al., 2002], [Mitra, 2003], [W3C, 2003]. WSs ensure systems interoperability based on platform and language independence by hiding implementation specifics behind a unified standard interfaces [W3C, 2003]. In Figure 1 (case 2) we show how the scenario (Figure 1, case 1) changes if the WSs are put into use. Creating of only a single type of adapter for all systems and the registry in the form of a WS interface is undoubtedly easier; and it can be automated so that whenever new software is installed it would essentially take no effort to expose it as a WS for the enterprise internal maintenance needs. Involving the WS technology for systems configuration also eliminates the need to configure those adapters. The problems one would experience in employing WSs for providing the necessary glue among software for reconfiguration purposes are related to the immaturity of the technology.

[1] Basic scenario (1) adapted from [Jablonski, Petrov, 2004]

3. ENFORCING CONFIGURATION RULES AUTOMATICALLY

WS-enabled registries can be made an active participant in complex enterprise systems configuration if complemented with WS-based processes, known also as WS-flows [ReFFlow, 2004].
The combination of complete configuration data stored in a registry and functionality is instrumental for automating the process of systems management. However, changes in the data have to be communicated from the registry to all systems and vice versa. There are two way to propagate configuration rules. The first one is using the registry as a *centralized configuration data source*. In such a scenario the registry plays a *passive* role in the reconfiguration process and supports some simple notification functionality that helps to inform users, systems and administrators of changes in configuration settings and possible conflicts. Systems administrators however would still have to intervene to reconfigure systems and resolve possible conflicts.

We envisage a more *active* role for the enterprise registries with regard to system configuration. In addition to identifying dependencies and resolving them upon configuration data changes the registry has to communicate these changes and the result of conflict resolution (also change in data) to all affected systems, receive feedback from those systems as a result of the reconfiguration, and react again if additional reconfiguration is necessary. Thus the registry performs actively in the reconfiguration process. So far such abilities have been enabled by the concept of Event-Condition-Action (ECA) rules [Cilia, Buchmann, 2002]. In a WSs world the ECA rules can be implemented in terms of complex compositions of tasks, i.e. WS-flows.

The logic necessary to implement the correct sequence of configuration activities could be extremely complex; moreover tangling it with the (registry) implementation is not a favoured approach. Similar problem has been addressed earlier in distributed computing and resulted in the development of the *workflow* technology as a complement to EAI solutions [Alonso, 2003]. Workflows are not only good in dispatching data among applications; they are also able to explicitly describe business logic and provide for task routing [Alonso et al., 2003]. In a WS-enabled environment the process-based approach brings even more advantages, for it reaps additional benefits from leveraging features inherent to the WSs. In a way the proposed approach extends an approach to dynamically reconfigurable platforms containing different systems [Jablonski, Petrov, 2004].

WS-flows are of great interest to the WSs community lately. A WS-flow is composite WS implemented using a process-based approach. That is, a WS-flow defines which tasks have to be executed, their execution order (control flow), and the exchanged data (data flow). All tasks or activities are performed by WSs. WS-flows can be written in any of the existing definition languages (e.g. BPEL4WS [Curbera et al., 2003] and BPML [Arkin et al., 2002]) and executed to perform the configuration logic on behalf of the registry. In Figure 2 we demonstrate how WS-enabled registries can be turned into active entities. The registry exposes a WS interface and participates in a WS-flow for the purposes of systems configuration. Installing a new application and, more precisely, uploading its configuration data on the registry is an event that starts a WS-flow instance execution. The process then performs reconfiguration logic as follows:

- Check the configuration settings of the application.
- Update configuration data for firewall depending on the configuration constraints. As a result the required port is either unblocked or not, or an alternative port number is appointed.
- Notify the firewall of the changes made.
- Update configuration data of the application at the registry – it is either allowed access via the port it asked connection through or not; if another port number is appointed the configuration data of the application is updated accordingly.
- Notify application of changes in its configuration data.
- Update users' data at the registry, specifying the new application they can use and the port number.
- Notify all users of the application that it can be accessed via a particular port.

This approach has some implications. Traditional workflows as well as WS-flows need a special process execution environment, called process engine. Hence, a process engine must be a part of the registry's architecture (Figure 2). If SOAP [Mitra, 2003] is used as a communication protocols, a SOAP processor is also necessary to process and route SOAP messages among all WSs. The sequences of activities defined by a WS-flow tend to be similar in different scenarios, mainly because of the relatively constant IT infrastructure of an enterprise. Here we recognise the necessity for WS-flows created especially for systems configuration, which we denote with the term *systems management rules*. Automatic creation of systems management rules is possible. Such WS-flows are a perfect playground for creating a collection of special *templates*

[Karastoyanova, Buchmann, 2004] that could be used to quickly produce reconfiguration WS-flows for particular system and scenario. Having such standardized templates for configuration WS-flows, even only for the internal use in an organization, has the potential to become a basic means to support system maintenance in an enterprise automatically.

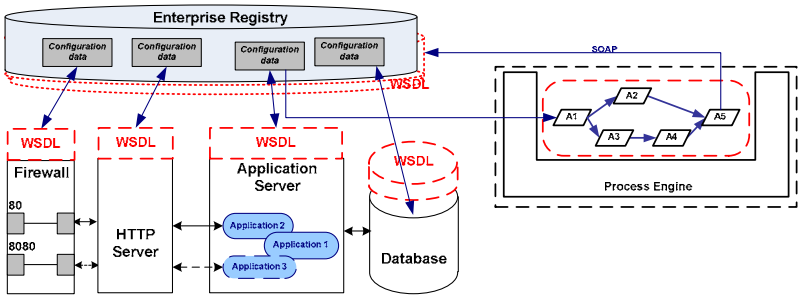

Figure 2. Implementing the complex configuration logic using WS-flows.

The success of the combination of enterprise registries and WS-flows in the process of system configuration is closely related to the progress in the development of the WS technology. The advantages of using WS-flows in systems management are substantiated by the *combined benefits of* using *registries* as central points of control [Jablonski, Petrov, 2004], *process technologies* for executing complex sequences of activities based on explicitly specified rules and the *WS* technology providing for seamless interoperability in a distributed heterogeneous environment. The systems management rules, and therefore the configuration WS-flows implementing them, are in a way a *point of convergence of* different *technologies*, and represent the potential for interplay of different paradigms in computing.

4. CONCLUSIONS

Nowadays enterprise systems configuration depends exclusively on the efforts of human system administrators. Central role in automating systems configuration can be assigned to a registry. A registry stores configuration data and rules in consistent manner. A WS-enabled registry can propagate changes in configuration settings to the affected systems in a seamless way. We advocate the use of WS-flows for governing the execution of complex configuration rules defined in a registry for the purposes of systems zero maintenance. To denote these configuration WS-flows we introduce the term “systems management rules”. This approach relies on supporting tools for creation of system management WS-flows from templates and the ability to customise them for use in the enterprise under consideration and its systems.

REFERENCES

Alonso, G. et al., 2003. *Web Services. Concepts, Architectures and Applications*. Springer-Verlag, Berlin Heidelberg.

Arkin, A. et al., 2002. Business Process Modeling Language”, *BPMI.org*.

Bellwood, T. et al., 2002. UDDI Version 3.0. IBM, HP, Intel, Microsoft, Oracle, SAP.

Cilia, M. and Buchmann, A. P., 2002. An Active Functionality Service for E-Business Applications. *ACM SIGMOD Record*, Vol. 31 No.1.

Curbera, F. et al., 2003. BPEL4WS 1.1. *http://www.ibm.com/developerworks/library/ws-bpel*

Jablonski, S., Petrov, I. et al., 2004. *Guide To Web Application & Platform Architectures*. Springer-Verlag, Berlin Heidelberg.

Karastoyanova, D. and Buchmann, A., 2004. Automating the development of Web Service compositions using templates. *Proceedings of “Geschäftsprozessorientierte Architekturen” Workshop, at Informatik2004*, Ulm, Germany.

Mitra, N., 2003. SOAP 1.2. W3C. *http://www.w3c.org/TR/2003/REC-soap12-part0-20030624/*

ReFFlow Project, 2004. *http://www.informatik.tu-darmstadt.de/GK/research.html*

W3C, 2003. WSDL 2.0 Part 1: Core Language. *http://www.w3.org/TR/wsdl20/*