Applying Web Service Compositions in Systems Management

Dimka Karastoyanova

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

Abstract: Systems management in any organization depends exclusively on the efforts of a system administrator. Every change in the IT infrastructure landscape requires changes in the systems configuration settings. We propose the use of enterprise registries for addressing the automation of systems configuration. Registries provide global view on the whole IT infrastructure of an enterprise, store a complete configuration data and maintain consistent record of systems state. The role of a registry in the automation of reconfiguration activities in an enterprise can either be passive - with the registry being only a data store for configuration data, or an active one. We view the active approach much more suitable for facilitating automatic system management. Active registries serve not only for data keeping but also command and perform reconfiguration directly onto the enterprise systems. To facilitate the seamless interaction between registry and all systems we propose to extend them with Web service capabilities. To define and perform all the complex sequences of tasks according to the configuration rules defined in a registry we advocate the use of Web Service compositions. We call these process-based Web services "systems management rules". Furthermore, the development of systems management rules can be automated by using Web Service Flows templates.

1 Introduction

Systems configuration and management is a time-consuming, tedious business that relies on the skills of a system administrator. The potential for automating it is great but it is not yet fully explored. The main reason is the considerably greater importance of creating software systems, integrating them, and automating the development of those systems, rather than paying attention and spending resources on solutions for the automation of the (re)configuration of the systems.

Managing software systems within an enterprise is a concern owing to the mix of software platforms, applications and resources an enterprise possesses. There are already existing approaches to integrating such heterogeneous environments. The integration effort, however, relates not only to providing for interoperability among system components in order to reach a certain goal. It rather involves a serious reconfiguration effort in the form of systems administration.

One of the ways to gain a global view on a whole enterprise is to logically centralize the control over it. A first step to do so is to develop central registry recording information about the enterprise. Since configuration is also a matter of control, it should also be centralized logically and the most appropriate approach to do so is to use a configuration registry as a sub-component of the enterprise registry. All systems of an enterprise store their configuration data in the configuration registry and rely on it. Based on the "information", which records the overall state of the enterprise systems, the administrators may define methods (rules, procedures) to control the whole system. This is why it can be said that an enterprise registry is a central point of control to the enterprise system. The data stored in an enterprise registry comprises application configuration data and configuration schema, notification interfaces, rules, authentication policies etc.

Clearly, an enterprise registry has to exhibit the following features: global scope, completeness, consistency, information model, resolution of dependencies, configuration rules enforcement. We dwell in detail on these characteristics and the goals aimed at by the use of registries in section 2.

Additional challenge is the degree of integration between the enterprise systems and the registry, which is a typical integration problem previously addressed by computing paradigms such as Middleware services, EAI, B2B, and the newest approach being the Web Service (WS) technology [Al03]. To facilitate systems configuration and management in an automatic manner we propose the use of a configuration registry equipped with WS capabilities (see section 3). WSs provide a unified way of exposing services via stable interfaces and benefit from the advantages of the ubiquitous and cheap medium, the WWW, used for communication. These WSs features simplify the interactions related to the automatic systems configuration and thus result in gaining centralized control over the whole enterprise systems configuration.

Using WSs for systems configuration addresses only a part of the difficulties in respect to automating this process. WS provide a standard way to integrate heterogeneous systems; but a registry has to store configuration rules and enforce these rules upon all dependent systems and thus reconfigure them accordingly. The sequence of steps necessary to do this might be simple and may require only a passive behavior on the part of the registry and its playing a role of a simple data source (section 4.1). However, having in mind that system configuration can be an extremely complicated matter there is a need to efficiently and explicitly describe the logic responsible for sequencing the reconfiguration steps in the correct order. This implies a more active role of the enterprise registry.

The experience shows that the best approach to explicitly define integration logic and task routing is a process-based one. We are convinced that WSs compositions are the most appropriate technology to apply (section 4.2). This technology combines the advantages of the traditional workflow technology and WSs. We conclude the paper and point out directions for further research in section 5.

2 Enterprise registries and systems configuration

Enterprise registries appeared as a result of the development of several technologies. Earlier work in this field includes directory service technologies such as LDAP [OL04, Jo98], X.500 [ISO90], and Active Directory [MS04], which mainly serve name and address resolution. Another group of registries – the middleware registries – store components and their interfaces, and are used basically for application discovery and dynamic binding. A different class of registries holds data about users and their privileges; these are usually applied as central points of authentication.

Registries exhibit several properties that motivate their use for systems configuration. While registries store various kinds of data, it is of great importance that they store relations among data, too. Completeness and consistency of the data and dependencies are ensured by the registry. Configuration data and their relationships can be used to define configuration rules. Registries in turn can be equipped with functionality to automatically enforce those rules on all affected systems in the enterprise. We discuss these features of registries in more detail next.

A registry stores information about the systems comprising the overall enterprise IT infrastructure. It has to manage the information and propagate any changes to all affected systems. One of the goals of a registry is to keep complete and up-to-date information about the whole enterprise. This goal is associated with several of the characteristics an enterprise registry supports, namely global scope, completeness, and consistency. An enterprise registry contains various kinds of *data*, including user information, authentication and authorization data, information about all software systems, applications and their components, the corresponding configuration data, the state of each system and application, as well as of the organization as a whole.

Having a complete and consistent global view on all system configuration data is certainly a proper approach towards automation of enterprise systems maintenance, because it implies a centralized control over the configuration data. Global view and control are instrumental for the additional goal of automating systems management. This goal is to help reduce manual intervention on behalf of administrators and shorten reconfiguration time in an enterprise.

Enterprise registries are active entities and therefore they contain also the *relationships* established among different kinds of information and information models, and rules governing those relationships. Based on these data and relationships registries keep the data complete and consistent with the existing rules, by resolving parameter conflicts and changing configuration data to comply with those rules. Registries track changes in configuration parameters and identify possible conflicts. More to that, registries are able to *interact with all enterprise systems*, resolve configuration dependencies and enforce constraints as a result of changes in the configuration data they hold; registries provide all enterprise systems with the appropriate feedback and information about environment changes. Therefore, pursuing the automation of system management and configuration brings in the necessity of information model that apart from the data model and rules defines also *functionality* it has to implement. Such functionality is generally related to *propagating changes* in the configuration data to applications in the system and thus reconfiguring those applications. Ordering the dependent reconfiguration activities properly also belongs to that functionality.

Propagating data changes to the systems in an enterprise calls for communication between the registry and the systems. We must be aware of the fact that the *communication* among the software systems inside an organization is impeded by the heterogeneous nature of the environment. Since communication among heterogeneous systems is a problem that requires integrating those systems, we find it suitable to WS-enable the enterprise registry by exposing its functionality as a WS. WSs provide for seamless interaction between a registry and the enterprise systems. This decision has significant advantages, which are discussed further in the next section.

We recognize two groups of mechanisms for *propagating configuration* decisions and changes to all systems. The first group requires a more passive role of the registry in the whole scenario. A passive registry appears to be only a source of complete configuration data, stored in consistent manner; no changes are allowed in the data stored by the registry itself, it is only capable of notifying the system dependent on the data of rejection or approval of a reconfiguration request (see section 4). The system administrator has still to intervene and reconfigure each single system.

A more active role is assigned to the enterprise registry if the second group of mechanisms is used for enforcing systems management rules. Should the active *rule enforcement* approach (see section 4.2) be taken advantage of, the degree of automation rises significantly. One of the mechanisms we recognize is the use of notification to inform systems of changes in their configuration data and perform the reconfiguration without any need for the administrator to do so manually. However the logic behind enforcing these rules is in most of the cases very complex to be enabled by simple programming routines executed by the registry. Therefore we introduce the use of WS-based processes to perform on the behalf of the registry. This approach not only facilitates the automation of data management in the registry, but also results in automated propagation of data changes and rules enforcement to the affected systems directly.

2.1 Basic scenario

The further discussion in this paper is based on the following scenario. In Figure 1 a simple infrastructure for Web applications is represented. It includes an application server, a database, an HTTP server and a firewall, and a registry containing data relevant to the configuration of the system, e.g. firewall configuration (ports on which HTTP access is allowed), application server settings, user information etc.

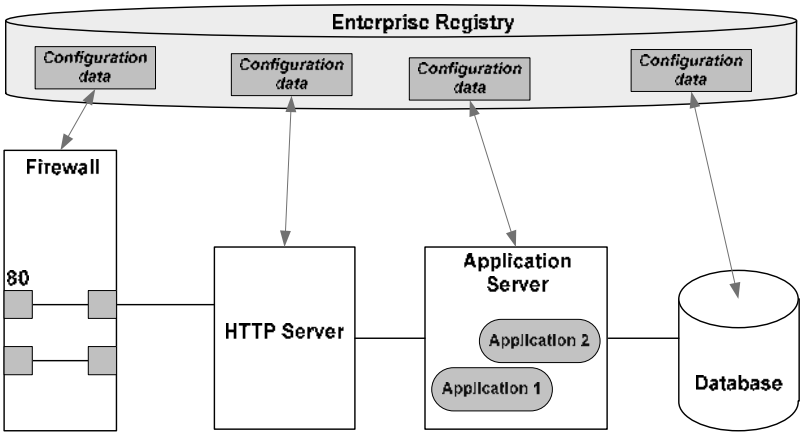

Figure 1: Basic scenario

All elements of this simple system have their configuration data stored into the registry. These data includes configuration files, configuration parameters, status data, and rules. The rules in turn consist of dependencies, relationships and constraints. Resolving dependencies (of system parameters) and conflicts, as well as maintaining completeness and consistency of configuration data are major functionalities a registry must support.

In the example in Figure 1 the firewall is configured to allow access to the applications on the application server only through port 80, and that information is represented at the registry. All other components of the infrastructure can query the registry and find out data about themselves and about all other components. But, in order to let all system components interact with and query the registry it must have the appropriate adapters at its disposal to ensure interoperability. This approach is rather inflexible; one can imagine the number of different systems, and therefore adapters, there is in an enterprise. This is a common problem faced by EAI technologies and is being addressed lately by the Web service technology [Al03]. The number of protocols and formats that a registry would have to support can be reduces by simply utilizing the advantages of the open, XML-based Web service standards. In the next section we shortly discuss the effects of utilizing WSs as a unifying approach towards facilitating the integration of the enterprise registry and all software systems.

3 WS-enabling enterprise registries

An enterprise registry has to interact with all the different systems in an enterprise in order to support their configuration. However, because of the heterogeneous nature of the enterprise IT infrastructure this interaction is only possible via multiple adapters that perform mapping between different protocols and data exchange formats. Developing the adapters is a hard work. It also consumes investment in terms of time for development, and resources. This is a problem pertaining to distributed heterogeneous systems and has been approached in different ways [Al03]. The most recent attempt is the WS technology. In the next figure we show how the scenario from the previous section changes when the WS technology is put into use.

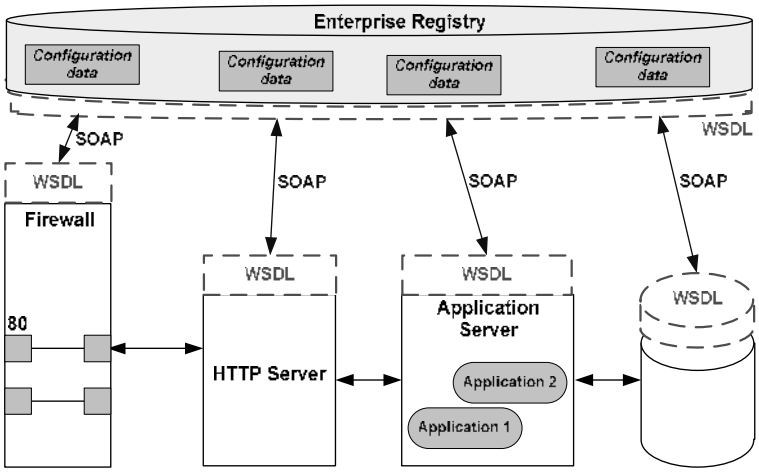

Figure 2: Applying WSs to promote interoperability and seamless integration

WSs follow the principles of service-oriented architecture (SOA). The technology ensures platform and language independence by hiding implementation specific characteristics behind unified interfaces; services are described using a single unified service description language (WSDL [W3Cb]). Communication among WSs is executed in terms of a messaging protocol, the SOAP [W3Ca], which specifies single message exchange format and utilizes HTTP. The technology also enables discoverability of services; the UDDI standard [Be02] is the one used to classify services and implement registries storing references to existing WSs.

Complementing enterprise registries with WS-capabilities has the *advantage* of making the need for all the different system adapters irrelevant (see Figure 2). The creation of only a single type of adapter for all systems and the registry in the form of a WS interface is undoubtedly easier. Besides, there are already tools available to automatically generate the WSDL interfaces so that whenever new software is installed it would practically take no effort and time to expose it as a WS for the internal configuration needs of the enterprise.

Moreover, WSs are based on open standard protocols which guarantee *interoperability* and at the same time foster flexibility behind the interfaces (i.e. the implementation of functionality is up to the service provider and can be based on any technology). In general, any applications, running on any platform can be exposed as a WS for the purposes of automated systems management. Involving the WS technology in this context does not only reduce the number of necessary adapters and protocols to be supported, but rather it eliminates the need to configure those adapters. Furthermore, such a solution is most appealing to small and medium-size enterprises, too, because of the cheap communication medium it requires, namely the WWW.

The WS technology still lacks lots of features [KB03, Al03] that would qualify WSs as a mature middleware technology. The *problems* one would encounter when employing WS for providing the necessary glue among software for configuration purposes are based exclusively on this fact. Among the missing features are notably security and transactional support, which we classify as instrumental for ensuring consistency and correctness of configuration data. Additionally, in the WS technology there is no standard way to specify access rights of users and applications to configuration data; authorization policies cannot yet be defined. These are some of the areas undergoing research in the field of WSs.

4 Enforcing configuration rules

A WS-enabled registry is able to interact with all enterprise systems smoothly using standard unified protocols and interfaces. The use of a registry in systems management has yet another objective – to resolve dependencies and conflicts of systems configuration parameters. Meeting this objective implies enforcing the configuration rules stored in the registry and propagating those rules by commanding reconfiguration to all affected systems. In this respect, a registry could be a passive participant in system configuration and management, but it is possible to assign it with a more active role in this process. These two points of view on the role of a registry in system management are discussed next. Ultimately, we would like to present the idea that WS-enabled registries can be made an active participant in systems configuration in a complex enterprise infrastructure. We propose to complement registries with service compositions that are the best way to explicitly define configuration tasks and their sequencing, and thus impose complex configuration rules.

4.1 Passive approach to systems configuration and maintenance

All departments in an organization have their own registries holding information about their employees, IT infrastructure, and probably about configuration. The idea of a centralized enterprise registry containing configuration data provides the administrators with a *global view* on the whole, and this is of great importance. Having a global view on all systems helps to solve dependencies among configuration parameters and resolve possible conflicts among systems related to their configuration.

Take for example the firewall scenario from above. Whenever a new application is installed it might require the firewall to provide access to this application through a port forbidden at that moment. What an administrator would do is to install the applications and save its configuration data into the registry, thus making it visible in the global picture. Then one would reconfigure the firewall, i.e. let the port free for authorized users only, and update the firewall configuration data at the registry. Reconfiguring a single firewall after a single application has been installed is an easy task. However, automating the (re)configuration of a whole lot of firewalls, applications and resources in an enterprise infrastructure with lots of users inducing changes in the system is a complex task.

The simplest step towards automating rules enforcement is to make the registry itself track all the changes in the system and inform the users of the application that, for example, the port can not to be used and thus reject the request. In this scenario the registry plays a *passive* role in the reconfiguration process. It is just a global container of *configuration data* and supports some simple *notification* functionality that helps to inform users, systems and administrators of the configuration settings and possible conflicts. The system management however should be done as usual by the humans. For the new application to work one has to change the firewall settings if of course such a change does not threaten the enterprise security policies. We envisage a more *active* role for the enterprise registries, which is much more beneficial with regard to system configuration. This idea is presented next.

4.2 Active approach to systems configuration – complementing registries with configuration WS-flows

We can induce a certain degree of automation into the scenario from above by making the registry to perform some logic for identifying dependencies, resolving and propagating them. For instance, whenever an application is installed the registry has to be able to recognize by inspecting its configuration data that this application requires access through a currently forbidden port on the firewall. The registry has to change the data it holds for the firewall configuration and propagate this new settings to the firewall. Additionally, it has to notify the application and all its users about the port to be used. The implication of this is that a registry not only holds information about system configuration (data model), but rather it can interpret its meaning and enforce the rules specified by this data by performing appropriate activities (information model). Further, the registry must be equipped with the ability to identify changes in the configuration data (performed by users, systems or administrators) and react appropriately.

Such behavior is usually described by the concept of Event-Condition-Action (ECA) rules [CB02] and is known from the research in adaptive databases. It requires inserting logic into the registry, so that it can perform the sequence of relevant actions upon event occurrence. The event-based systems are known to be the most efficient in such scenarios [Al03]. However efficient and excellent those systems are in detecting and reacting to events, it is very important how the logic specifying the sequence of actions in reaction to an event is defined.

The logic necessary to implement the correct sequencing of (re)configuration activities could be extremely complex. As the experience in various computing fields shows, tangling orchestration logic with the (registry) implementation is not the best approach. Similar problem has been approached earlier in the development of distributed computing and resulted in the development of the workflow technology to complement the EAI solutions (message brokers and so on). Workflows are not only good in dispatching data among applications [LR00] (administrative workflows, document routing and processing); they are able to explicitly describe the business logic and provide for task routing. In a WS-enabled environment the process-based approach brings even more benefits, by leveraging the advantages inherent to the WSs technology.

WS-based processes are a subject of particular attention in the WSs community lately. WS-based processes are complex WSs, also called composite WSs [Al03] or Web Service Flows (WS-flows) [Ka03]. A WS-flow is a business process that specifies a collection of tasks to be performed for achieving a certain goal, and defines the conditions under which tasks execution is ordered and data is exchanged, control and data flow respectively. All tasks or activities defined by a WS-flow are performed by WSs. There are already two industry-driven specifications of WS-based process definition languages, BPEL4WS [Cu03] and BPML [Ar02]; either of them has the potential of becoming a standard. WS-flows and the corresponding definition languages can be used to describe the configuration logic that has to be executed by the registry – they are suitable for describing explicitly configuration logic in a WS environment. In fact, the WS-flow running on a process engine will execute the configuration logic and thus enforce the rules, update registry configuration data, notify all participating systems and so on, on behalf of the registry.

In the following example the context of the scenario from the previous section remains the same, i.e. all participating systems are exposed as WSs. The registry itself exposes a WS interface and is a participant in a WS-flow for the purposes of configuration. Installing a new application and uploading its configuration data at the registry is an event that starts a WS-flow instance at the process engine (see Figure 3). The process then performs all the configuration logic as follows:

- Check the configuration settings for the application.
- Update configuration data for the firewall but follow the configuration constraints. As a result the required port is either unblocked or not, in some cases an alternative port number could be appointed by the firewall to this application.
- Update configuration data of the application at the registry based on the result of the previous step.
- Notify the application of changes in its configuration data – either it is allowed access through the port it required or not; or if another port number is appointed the configuration of the application is updated accordingly.
- 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 at a particular port.

Each time configuration data is updated at the registry the respective system is reconfigured accordingly.

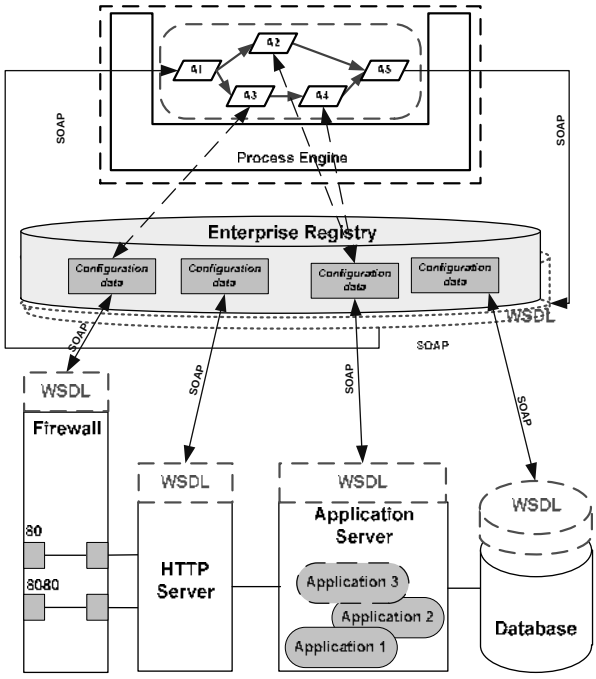

Figure 3: Implementing complex systems configuration logic using WS-flows.

Although this is a very simple example it requires a whole bunch of activities to be performed in a certain order and has several decision points. In a more complex scenario the number of activities rises significantly. To cope in an automated manner with the complexity of the logic imposed by the configuration rules a process-based approach is employed. Using such an approach helps to both orchestrate the complex sequences of tasks and follow all rules; that is what workflows are good at [Al03], after all. A WS-enabled enterprise infrastructure implies the use of WS compositions. WS-flows are capable of handling the complex logic for resolving conflicts and enforcing configuration rules. They can save lots of administration work and maintain the global view on the systems and their state consistent.

To be able to leverage the capabilities of WS-flows we need to take into account the following implications. Traditional workflows as well as WS-flows require a special process execution environment, called process engine [WC04, IBM02]. Such an engine must therefore be a part of the registry's architecture (see Figure 3). A SOAP processor is also necessary for processing and routing SOAP messages among WSs, if SOAP is the protocol chosen for communication at all; different communication protocols can also be used [Vi04].

Just like automating systems configuration in an enterprise, automating the creation of WS-flows for the purpose of systems management is equally important. The sequences of activities defined by WS-flows tend to be similar in different scenarios, mainly because of the relatively constant IT infrastructure of an enterprise and the small variety in the nature of the configuration activities. Drastic changes in the IT infrastructure of an enterprise are rare, and therefore the business processes in this application domain are relatively small in number and in diversity as compared to business processes applied in other more dynamic domains. Based on this fact we recognize the necessity for WS-flows created especially for systems management. We call these configuration WS-flows *systems management rules*. Because of the small variety of systems management rules the principle of reusability can be much more easily and successfully applied to generate the WS-flows descriptions automatically and on demand. The WS-flows implementing these rules are a perfect playground for creating a collection of special-purpose *templates* that could be used to quickly create WS-flows [KB04] for the purposes of reconfiguring particular set of systems for a particular scenario. Having such standardized templates for systems management rules, even only for the internal use in an organization, has the potential to become a main tool for automating systems configuration and management in an enterprise. There is a possibility to generate automatically a WS composition definition from a template, even at run time of the WS-flow implementing the rule [KB04].

The success of enterprise registries in combination with WS-flows in systems configuration is closely related to the progress in the development of the WS technology. Furthermore, this approach would only gain acceptance if there are tools available to support the generation of system management rules (i.e. WS-flows) both at their modeling and execution time. The advances in the field of WS compositions will reduce the work of an administrator from manual reconfiguration to just defining rules for systems management.

The advantages of using WS-flows for configuration are substantiated by the combined benefits of using:

- *registries* as central points of control,
- *process technologies* for describing and executing complex sequences of activities based on explicitly specified rules, and
- *WS technology* providing for seamless interoperability in distributed heterogeneous environment.

The configuration WS-flows are in a way a point of *convergence of different technologies*. They exhibit the potential for interplay of different paradigms in computing.

5 Conclusions

Nowadays enterprise systems (re)configuration depends exclusively on the efforts of human system administrators. Configuring and managing systems is a process that can be automated. Central role in automating systems configuration can play a registry storing configuration data and rules, and instrumented with functionality for resolving dependencies among IT systems and especially among their configuration parameters, obeying constraints, and enforcing rules. Apart from storing data and rules a configuration registry must be able to track changes in configuration data and propagate the reconfiguration to the affected systems and applications. Some examples of events resulting into systems reconfiguration are installing a new application, request for outgoing connection to a firewall, change in user settings etc.

In this paper we presented the idea of enabling an enterprise configuration registry with WS capabilities. Providing a registry and the systems in an organization with WS interfaces allows them to interoperate in a standard way. Thus propagation and enforcement of systems reconfiguration rules is simplified. To promote zero maintenance, we push the idea further and extend the WS-enabled registry with the ability to explicitly define complex configuration logic by applying a process-based approach. We promote the use of WS-based composition, WS-flows, that can govern the execution of complicated configuration rules. We denote such configuration WS-flows with the term “systems management rules”. The creation of such compositions can further be automated with the help of standardized templates for WS-flows. The successful application of this approach depends on the availability of tools for creation of WS-flows for system management from templates and the ability to customize them for use in the enterprise under consideration and its systems.

References

[Al03] Alonso, G. et al.: Web Services. Concepts, Architectures and Applications. Springer-Verlag, Berlin Heidelberg New York, 2003.

[Ar02] Arkin, A. et al.: Business Process Modeling Language, BPMI.org, 2002.

[Be02] Bellwood, T. et al.: UDDI Version 3.0, IBM, HP, Intel, Microsoft, Oracle, SAP, 2002. http://uddi.org/pubs/uddi_v3.htm

[CB02] Cilia, M.; Buchmann, A. P.: An Active Functionality Service for E-Business Applications. ACM SIGMOD Record, Vol. 31 No.1, March 2002.

[Cu03] Curbera, F.et al.: Business Process Execution Language for Web Services (BPEL4WS) 1.1, May 2003, http://www.ibm.com/developerworks/library/ws-bpel

[ISO90] Data Communications Network Directory: ISO 9594, Recommendations X.500-X.521, 1990.

[IBM02] IBM AlphaWorks: IBM Business Process Execution Language for Web Services JavaTM Run Time (BPWS4j), IBM, 2002. http://www.alphaworks.ibm.com/tech/bpws4j

[Jo98] Johner, H. et al.: Understanding LDAP. IBM International Technical Support Organization, 1998. http://www.redbooks.ibm.com

[Ka03] Karastoyanova, D.: “Creation and Deployment of Web Services and Web Service Flows”, A Tutorial, In (Kotsis, G., Bressan, S., Katania, B., Ibrahim, I.K, Eds.): Proceedings of the 5th Int. Conf. on Information Integration and Web-based Applications and Services (iiWAS2003), Austrian Computer Society, September 2003.

[KB03] Karastoyanova, D.; Buchmann, A.: “Components, Middleware and Web Services”, In (Isaias, P., Karmakar, N., Eds.): Proceedings of IADIS International Conference WWW/Internet 2003. IADIS Press, 2003; Volume II, pp. 967-970.

[KB04] Karastoyanova, D.; Buchmann, A.: A Procedure for Development and Execution of Process-based Composite Web Services. Poster presentation, to appear In Proceedings of International Conference on Web Engineering, ICWE 2004, Munich, July 2004.

[LR00] Leymann, F.; Roller, D.: Production Workflow. Concepts and Techniques. Prentice Hall Inc., 2000.

[MS04] Microsoft Corporation: Microsoft Active Directory Architecture, White Paper, http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/activedirectory/deploy/projplan/adarch.mspx, 2004

[OL04] OpenLDAP: http://www.openldap.org/, 2004.

[WC04] Workflow Management Coalition (WfMC): www.wfmc.org

[W3Ca] World Wide Web Consortium (W3C): SOAP Version 1.2. W3C Recommendation, 2003. http://www.w3.org/TR/soap12-part0/

[W3Cb] World Wide Web Consortium (W3C): Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Working Draft, 2003. http://www.w3.org/TR/wsdl20

[Vi04] Vinoski, S.: “Extending Web Service Bindings for Real-world Enterprise Computing Systems”, In (Bevinakoppa, S., Hu, J. Eds.): Proceedings of the 2nd Int. Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI 2004), April 2004. INSTICC Press, Portugal, pp. 3-8.