A KNOWLEDGE-BASED SYSTEM FOR THE VALIDATION OF THE DEPLOYMENT OF SOFTWARE UNITS

Fco. Javier Blanco, Laura Díaz-Casillas and Mercedes Garijo
Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Madrid, Spain

Keywords: Knowledge-based systems, Validation, Software deployment.

Abstract: Today, many business applications are developed following SOA principles. One of the activities required for their implementation is deployment, a complex process that usually is done by hand, being necessary to develop new tools to facilitate it. This article presents a knowledge-based system that validates the deployment of software units on a particular environment, before executing them. The system is based on an information model and has been implemented with Drools 5.0 and as an OSGi bundle to be integrated into a deployment and configuration architecture.

1 INTRODUCTION

Nowadays, many business applications are developed according to a service-oriented architecture (SOA) (Hashimi, 2003), which promotes their availability, reliability and scalability. One of the operations to execute during the development of such applications is the deployment of services, in which all the activities required for the execution of these services are carried out. A service is a software system that provides a particular function to a user or to another system. Usually, services cooperate with each other to achieve a common goal, being necessary to check their availability. Also, the requirements demanded by the services have to be considered in order to select the most suitable nodes for their deployment. It is also necessary to take into account that during the development of the application, services and environment characteristics can change, being necessary to modify deployment activities or, at least, validate them. Therefore, deployment is a complicated process. However, it is a task that is usually done manually, having a high cost associated and being one of the main sources of error during the systems development.

Due to several problems that have to be solved, automation of deployment is still a research area. ITECBAN is an innovation project funded by the Spanish Ministry of Industry, Tourism and Trade through CENIT program. Its main objective is to create a platform that serves as a basis for developing core banking information systems. In this context, a deployment and configuration architecture (from now on DCA) has been developed to support deployment activities of software units on different environments. This architecture uses a resource model explained in (Cuadrado et al., 2009), based on the model presented in (Ruiz, 2007), which is suitable for distributed and heterogeneous environments. This model is also based on the CIM (DTMF, 2005) application model and the standard of components and distributed systems deployment established by OMG (OMG, 2006). Specifically, the information model is divided into three models dedicated to the definition of: the environment to be used on the deployment (Deployment Target), the software units to deploy (Deployment Unit), and the deployment plan that determines the activities to be carried out (Deployment Plan).

The use of deployment plans improves the automation of the process. Each deployment plan has an objective, consisting of executing an operation over a software unit in a specific environment. Usually, the fulfillment of a plan requires to add a set of previous operations, in order to satisfy the requirements of the unit to deploy. But it is possible that since the creation of the plan until its execution, the state of the environment changes, being necessary to verify that the operations in the plan are still suitable and do not conflict with available resources and already deployed units in the environment.

According to Binder (1995), who illustrates how it is practically impossible to do effective software test-

Blanco F., Díaz-Casillas L. and Garijo M. (2010).
A KNOWLEDGE-BASED SYSTEM FOR THE VALIDATION OF THE DEPLOYMENT OF SOFTWARE UNITS.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence - Artificial Intelligence*, pages 305-310
DOI: 10.5220/0002726303050310

ing and validation without automation (Binder, 1995), such verification has to be carried out in an automatic way. There are different approaches to automate software validation and verification. Playle and Beckman (1996) compare people who make decisions, neural networks, and rule-based programs, drawing the conclusion that human decisions are more precise but require more time than machines; neural networks, with nonlinear rules, are relatively unreliable in making the decisions; and rule-based machine programs are very fast and almost as accurate as the humans (Playle and Beckman, 1996).

This article presents an automatic system to validate deployment plans. From the information contained in the plan and the current state of the environment, the system evaluates if the activities of the plan are appropriate for achieving the objectives set at its creation. Therefore, the purpose of the validation system is to ensure that the plan is suitable for the selected environment before executing it. Validation results are shown to the user, avoiding potential problems and helping to solve actual or future errors. Results indicate if the plan is valid, or contains operations that can cause errors, or should be aborted. For its implementation, a knowledge-based system has been used, it means, a system that tries to solve problems through reasoning and heuristic methods. This approach is often used to emulate the behavior of an expert in a particular area of expertise, being known as expert systems.

The rest of the article is organized as follows. Section 2 exposes a description of the implemented system, indicating the elements that compose its architecture and their relationship with the system in which has been integrated. Then, section 3 explains technologies used for its implementation, and section 4 shows results obtained after testing the system. Section 5 presents related work and finally, section 6 shows the main conclusions and possible future lines of work.

2 SYSTEM DESCRIPTION

To define a deployment process, the user selects a software unit and a target environment. As a result, a deployment plan that includes all required activities is created. But, before its execution, it is necessary to verify that the plan is correct. This is the purpose of the validation system, which analyzes the plan to execute and the state of the environment selected for its deployment, and generates a report showing the user the most appropriate actions to be performed.

The system is integrated into the DCA (Ruiz et al., 2008), whose main components are the Execution Manager, the Environment Manager, and the Deployment Repository. Execution Manager is responsible for activating the validation immediately before launching the plan, indicating the information contained in the deployment plan. The validation system is integrated into the Environment Manager that provides the current state of the environment. To carry out the validation, the system has to access the Deployment Repository through Web services in order to get the information of the software units.

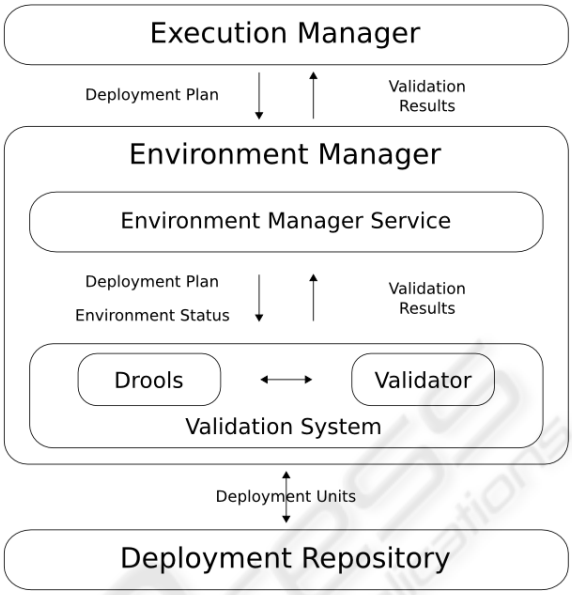

Figure 1: Validation System.

Figure 1 shows the exchange of information between systems in the architecture (DCA).

The implemented system analyzes every activity in the deployment plan, considering possible dependencies between them, and verifying if all requirements for the deployment are fulfilled. In some cases activities will have to be aborted, while in others it will be enough to warn the user about possible conflicts that can arise from its execution.

2.1 Available Knowledge

Initial knowledge consists on a deployment plan, i.e. the set of activities to be performed, and the state of the environment where the deployment will be carried out.

Each activity involves the execution of an operation (installation, start, update, uninstall or stop) of a software unit on a container belonging to a node in a particular environment. Users create deployment plans with the aim of working on a software unit, but units have constraints and dependencies with other

units, and if they are not currently satisfied in the environment, new activities have to be added to the plan to fulfill all the necessary requirements. Specifically, each unit exports a set of resources and has dependencies with other units, which export resources required for its execution. Moreover, units have constraints that need to be fulfilled by the container in which the deployment will be carried out (e.g. EJB container, glassfish server,...). Deployment unit data are stored in the Deployment Repository.

The information to be considered depends on the activity to be performed on a software unit. For example, to start a unit requires to verify that all dependent units are available, while to stop a unit needs to consider if this unit is being used by others. To manage such information, two different dependency graphs are used, one manages the units required by a unit, it means its dependencies, while the other analyzes the deployed units that are using and are used by a unit, it means the relations among runtime units in the current environment.

2.2 Knowledge-based System

Using a knowledge-based system for the validation of the deployment of software units allows domain expert users to define the checks to perform during the validation and the most appropriate actions to follow in each case. Inference engine takes this information, stored in the knowledge base, to verify the operations of the deployment plan that is going to be executed, considering the requirements of the software units involved in the plan and the current state of the environment related to that plan.

A rule-based system has been used to specify conditions and actions to perform under specific situations. Rules are represented in a decision table, enabling their updating in the future in an easy way. Table columns contain the conditions and associated actions to execute in each case, determining the system behavior. Therefore, each row of the table corresponds to a rule, that are classified into several groups:

- Initialization, this rule starts the verification of the deployment environment related to the plan.
- Environment elements, this set of rules verifies that the nodes and containers of the deployment environment are in a correct state.
- Deployed units, these rules check if the unit associated to an activity is already deployed on the environment and analyzes its state.
- Necessary units, in this case, rules are used to check if dependencies imposed by the unit are satisfied by the environment. For example, if a unit that depends on another is going to be started, it is necessary to verify if the required unit is currently launched.
- Overversion units, these rules check if there are other units with the same name but different version already deployed on the environment.
- Completion, these rules end the validation process.

The execution order of the rules corresponds to the order in which they are exposed, for example, if a container is not available (feature considered in the environment elements, the second group of rules), the validation ends and a report is created to inform the user about this fact.

2.3 Results

The result of the validation of each activity is composed of a list of messages, each of them contains the type of the response and a text, which explains the reason and the cause of the response. In particular, possible answers are:

- NONE indicates that the outcome of the validation has been positive and the activity can be executed.
- NEXT, in this case, the result of the activity to perform is already implemented in the environment, so this activity is not necessary to be carried out.
- WARNING indicates that there is a conflict and the execution of the activity can cause problems.
- ABORT implies that there is a problem in the environment and the associated activity can not be executed.

For example, if one of the plan activities consists of launching a deployment unit that is not present in the container, validation system will report an ABORT message, in which the text will indicate that the unit is not present in the container, showing also the data related to that unit.

If an activity has to be aborted, all activities that depend on it will generate an ABORT response too.

3 IMPLEMENTATION

The validation system has been developed as an OSGi bundle. OSGi (OSGi Alliance, 2009) is a Java-based service platform that implements a dynamic component model. Each component or bundle is a dynamically loadable set of classes, configuration files, and resources, that hides its implementation details and

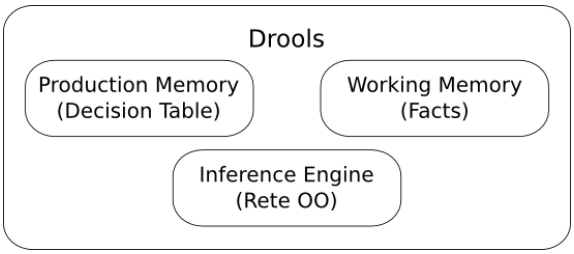

Figure 2: Drools components.

declares its dependencies to other components. Communication between them is established through services. And a service registry allows bundles to know available services in each moment. Therefore, our system acts as a dynamic module that can be connected and disconnected to the global architecture without requiring a reboot. It provides a validation service that can be used by any other module. These modules only have to indicate the necessary arguments and will receive a standard respond with the validation results.

The use of Spring Dynamic Modules (DM) for OSGi Service Platforms (SpringSource Community, 2009) has facilitated to build a Spring application that comply with OSGi standard.

Rule-base reasoning system has been implemented through Drools Expert, a component of Drools (JBoss Community, 2009). It is a business logic integration platform developed by JBoss Community. There are several suitable options for implementing an expert system such as Jess, Mandaraz, and OpenRules. But Drools Expert has been selected because:

- It is an open source product with a great support community.
- It works with decision tables in a user friendly format, using a spreadsheet software, such as OpenOffice Calc or Microsoft Excel.
- It is completely correlated with Java, allowing to import and use Java classes in the rules.
- It supports the inclusion of the entire framework on a Maven bundle to be executed as an OSGi service, which offers the ideal features to be developed within DCA.

Drools Expert components used in the validation system are presented in Figure 2.

It has been necessary to update the bundle provided by Drools to version 5.0 from the implemented version 4.7.1. In addition, specific libraries have been added to allow the use of a decision table in a spreadsheet format for managing the rules and to fully support MVEL and Java rule languages.

Furthermore, a Web interface has been developed to interact with the validation system. It has been implemented as a war bundle, which uses Spring DM to access to the system through the Environment Manager, and Spring MVC for data management and presentation. For its implementation, a controller form has been used, which collects data introduced by the users and manages the displayed information and the navigation between different views, implemented through JSP pages.

4 TESTS

A set of tests has been carried out to check the performance of the developed system. Firstly, unitary tests were executed, using different deployment plans over a test environment. Activities of the plan, software units, resources and nodes were configured according to several situations to get a great quantity of different test cases, generating multiple test deployment plans to validate all the rules. Validation system was executed for each generated plan. Table 1 presents the results of some performed test.

Later, the system was integrated into the Environment Manager within DCA and integration tests were carried out.

Finally, the Web interface was used to test the system with deployment plans generated by expert users in a real environment.

5 RELATED WORK

Most of the automated validation and verification of the deployment of software units is related to computational grids or grid computing. This technology enables the integrated use of remote high-end computers, databases, scientific instruments, networks and many other resources to execute the pieces of a program in parallel, having many similarities with SOA. According to Lacour et al. (2005), application deployment is responsible for discovering and selecting resources, verifying and validating them, placing applications on computers and finally launching application processes. Several ways are presented to describe applications, suggesting different deployment planners for each one. But defining a planner, with its planning algorithm and validation, for every type is too costly. Therefore, a generic application description model is proposed to have only one planner. In this way, specific and heterogeneous software units are translated into generic deployment units according to the model to facilitate their management (La-

Table 1: Activity validation examples.

Scenario	Result Type	Result Message
Installing a software unit in a container in which a same version unit has already been installed	Next	Activity already done
Launching a software unit A in a container in which has already been installed. Also, unit A depends on a resource exported by unit B, and this unit has been correctly installed and launched (due to previous activities in the same or in other deployment plan). Both units fulfill locality requirements	OK	Activity can be executed
Installing a software unit in a container that is not accessible	Abort	Container is not accessible
Installing a software unit in an environment in which a higher version of the unit is already installed	Warning	A higher version of the unit is already installed
Stopping a software unit required by another unit that is already being executed	Abort	This unit is being used by another unit
Uninstalling a software unit required by another unit that is stopped. No other unit satisfies this dependency	Warning	This uninstallation produces that a dependent unit could not be launched

cour et al., 2005). Our validation system is also based on a generic information model, enabling to use the same system with different software units.

Cannataro and Talia (2003) propose the use of grids to support Parallel and Distributed Knowledge Discovery (PDKD) systems. These platforms contain a combination of large data sets, geographically-distributed services, data, users and resources, and a computationally intensive analysis. PKDP execution plans contain graphs describing the interactions and data flows among data sources and are used and stored in a repository. Also, a resource allocation and execution management service is used to find a mapping between an execution plan and available resources, with the goal of satisfying requirements and constraints. In this context, execution plans are also responsible for managing and coordinating the application execution (Cannataro and Talia, 2003). Our system uses graph to validate resources, requirements and constraints. The validation is always done before executing the plan, ensuring its performance. Moreover, our system is published as a service, not as a group, being able to be launched in parallel with other services due to OSGi features.

Another area related to automated validation and verification of the deployment is to solve real-world planning problems. Wilkins and desJardins (2000) propose the development of methods based on rich knowledge models in order to make planning systems useful for complex environments. The goal of planning is not only to build a plan, it also has to be validated to be used in the environment. They expose that realistic planning systems must support execution validation and plan modification during execution (Wilkins and desJardins, 2000). Our system performs the verification and validation before the deployment, avoiding that the environment reaches an invalid state. Moreover, we have considered plan modifications before its execution to avoid the abovementioned problem as future work.

In (Romano and Palmer, 1998), a decision support system is developed to aid the validation and verification of requirement sets and to enhance the quality of the resulting global design by reducing risks. This system is used to detect four major types of potential risks: ambiguity, conflict, complexity and technical factors. In our system, ambiguity and conflict are solved by the use of a generic model that describes the deployment plan and the requirements of each unit. Complexity is solved by the use of an automatic system (machines treat any term with indifference if they are well programmed). Therefore, to validate technology factors is the objective of our system.

6 CONCLUSIONS

Enterprise applications deployment is a complex task that requires new tools to facilitate its development. This article proposes the use of a knowledge-based system that validates the deployment of software units on a particular environment. The system is integrated into a deployment and configuration architecture and based on its information model, which represents deployment units, plans and environments.

Deployment process begins with the creation of a plan, whereby the user determines the deployment unit on which he wants to perform an operation in a specific environment. To avoid that possible changes in the environment cause errors in the execution of a

deployment plan, an automatic validation system has been developed to advise users if the deployment process will not be able to perform correctly and why. After analyzing the unit requirements of each plan activity, it checks the current state of the environment, including existence, location and state of nodes, containers, and available resources over deployed units. A knowledge-based system has been developed due to these systems offer more facilities than conventional software systems to collect, interpret and use their knowledge, helping human experts to solve problems.

The validation system is presented as a dynamic and independent OSGi module and its implementation is based on Drools 5.0. Rules have been represented by a decision table, in spreadsheet format, facilitating their understanding and therefore, their update in the future. After the implementation of the system, a set of tests has been carried out to verify its performance.

Some improvements that can be developed within the validation system are:

- Currently, the system validates installation and uninstallation plans, being possible to extend the system to verify configuration plans.
- Nodes, containers and deployed units in the environment are analyzed by the system, but also resource properties could be examined to increase the security level during the execution of the plan.
- Also, algorithms to perform an optimal deployment of units, analyzing required and available resources in the environment, could be developed to improve the performance of the system.

ACKNOWLEDGEMENTS

We would like to express our appreciation to the Spanish Government for funding this project under the IT innovation project ITECBAN (MITYC CDTI-CENIT 2005). We would also like to express our appreciation to José Luis Ruiz, Félix Cuadrado, Rodrigo García, and Marco Antonio Prego for their help to develop the validation system.

REFERENCES

Binder, R. V. (1995). Testing Object-Oriented Systems: A Status Report. www.stsc.hill.af.mil/crosstalk/1995/04/testinOO.asp.

Cannataro, M. and Talia, D. (2003). The knowledge grid. In *Communications of the ACM*, volume 46, pages 89–93.

Cuadrado, F., Dueñas, J., García, R., and Ruiz, J. (2009). A model for enabling context-adapted deployment and configuration operations for the banking environment. *Networking and Services, International conference on*, 0:13–18.

DTMF (2005). Common Information Model (CIM) 2.1. www.dmtf.org/standards/cim.

Hashimi, S. (2003). Service-Oriented Architecture Explained. O'Reilly Media, Inc.

JBoss Community (2009). Drools - Business Logic integration Platform. jboss.org/drools.

Lacour, S., Perez, C., and Priol, T. (2005). Generic Application Description Model: Towards Automatic deployment of applications on computational grids. In *Grid Computing, 2005. The 6th IEEE/ACM International Workshop on*.

OMG (2006). Deployment and configuration of component based distributed applications version 4.0. OMG documents formal/06-04-02.

OSGi Alliance (2009). OSGi - The Dynamic Module System for Java. www.osgi.org.

Playle, G. and Beckman, C. (1996). Knowledge-Based Systems and Automated Software Validation and Verification. www.stsc.hill.af.mil/crosstalk/1996/07/knowledg.asp.

Romano, J. and Palmer, J. (1998). TBRIM: decision support for validation/verification of requirements. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 3, pages 2489–2494.

Ruiz, J. (2007). *A policy-driven, model-based software and services deployment architecture for heterogeneous environments*. PhD thesis, Departamento de Ingenieria de Sistemas Telematicos, Universidad Politecnica de Madrid.

Ruiz, J., Dueñas, J., and Cuadrado, F. (2008). A service component deployment architecture for e-banking. In *AINAW '08: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops*, pages 1369–1374.

SpringSource Community (2009). Spring Dynamic Modules for OSGi Service Platforms. www.springsource.org/osgi.

Wilkins, D. E. and desJardins, M. (2000). A Call for Knowledge-based Planning. *AI MAGAZINE*, 22:99–115.