Enterprise Application Deployment: A model driven approach

Peter Golibrzuch
Lufthansa Technik AG
peter.golibrzuch@lht.dlh.de

Alexander Holbreich
Lufthansa Systems AG
alexander.holbreich@lhsystems.com

Simon Zambrovski
Hamburg University of Technology
zambrovski@tuhh.de

Abstract: Today's enterprise applications are based on numerous interrelated components that capture domain-specific, generic or infrastructural functionality. Each component is based on several constituting artifacts where a certain artifact is also needed by different components. The dependencies between components and artifacts become even more complex when versioning is required. Additionally, a software development process requires several runtime environments for development, testing, consolidation and production for which elements of artifacts need to be configured accordingly. For managing the deployment of enterprise applications throughout an application's life cycle, a number of commercial tools are available. However, such tools often focus on specific technologies and require an integrated approach for build and deployment management. In this paper we present a different approach that allows for an agile model-driven deployment process that can be adapted to the desired level of detail. We propose the use of models to capture the complexity of interrelations between components, artifacts and their versions. By using either standard languages such as UML or domain-specific languages for the definition of system models, suitable tooling will be used to provide graphical user interfaces allowing for specifying required model elements and attributes.

1 Introduction

The context of the issues we would like to address refer to state-of-the-art enterprise applications that are commonly found in real world scenarios. The server part of the application implements business logic and persistence and is based on the Java 2 Enterprise Edition (J2EE) [JSR06]. The execution of business logic is either triggered by web service calls or by the presentation layer. The latter is packaged as a client part of the application which is delivered to client machines by Java Web Start [JWS06]. The client is based on the Rich Client paradigm and uses either Eclipse Rich Client Platform [ML05], [Ecl] or a proprietary Swing-based framework. In such a setup, one comes across several artifacts and components. In J2EE for example, descriptors are needed as additional artifacts for deploying an application on an application server. While descriptors are mainly used for the management of components, they also capture the configuration details of runtime environments such as specifying database connections or resource mappings for different

servers. This leads to a high complexity of dependencies and interrelations between artifacts and respective environments when deploying an application in a different runtime environment. The same holds for the client part of the application where dependencies of features and plugins are manifested by means of descriptors.

In common application life-cycles, focus on application operation, change management and quality assurance only begins once an application has left the development stage. Changing functionality of an application leads to different versions of components and artifacts that increase the complexity of dependencies discussed previously. For example, a particular component can only be used in combination with a specific version of another component. The dependencies of components need not only be considered when assembling an application but also need to be reflected in corresponding artifacts. Additionally, artifacts such as descriptors underlie versioning themselves, for example as results from changing a servlet mapping.

On an abstract level, components, artifacts and runtime environments are identified as domain entities which depend on each other and need to be considered during deployment. This includes the assembly of those entities according to their respective versions in order to ensure correct deployment. Managing the dependencies is seen as a constant activity throughout the application life cycle and is also a major cost factor in application change management.

In our opinion, existing approaches to deployment management do not satisfy the requirements of a deployment scenario as described in the example. We see Continuous Integration (CI) [DMG07] as a sophisticated and integrated approach to build and test management. We recognize the gain in efficiency and productivity by applying CI in software development projects, however, we believe that Continuous Integration focuses on building artifacts and components and is not a solution for complex deployment scenarios.

From a practical point of view, basic build tools such as Make [Her03] and Apache Ant [Ant] have proven to be successful aids for automating generic build processes as part of software development. Also, extensions of these basic tools such as Apache Maven [Mav] that include configuration and repository management have obtained global acceptance. However, we believe that the complexity of deployment scenarios with versioned artifacts is not manageable by means of build tools that express dependencies and interrelations through the use of text and property files.

Consequently, we integrated basic build tool support with an approach based on entity relationships for modeling the dependencies of components required for deployment. Our experiences lead us to believe, that such an attempt is only suitable for simple scenarios. With a rising interdependence of components and artifacts, an entity relationship model is not an adequate solution. Due to the complexity of dependencies, they can not be modeled as relations but as entities. This fact makes the usage of ER-models [Che76] unsuitable for the efficient representation of configuration data.

Therefore we propose model driven software deployment as a means to manage the complexity of the deployment process on the required level of abstraction by defining models and enriching the model with parameters. We also stress that our approach is applicable for small and mid-size software development projects as well as for project budgets that do

not allow for the use of full-fledged software development environments like MKS [MKS].

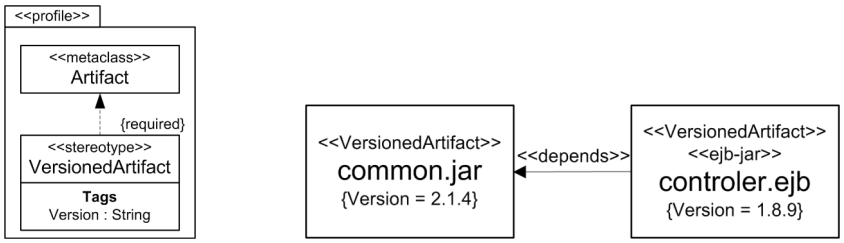
2 Model Driven Software Deployment

The approach that we propose uses a model as a means to express the dependencies of components, artifacts and corresponding versions. The approach is a specific case of Model Driven Software Development where the software development process is supported by model transformations towards a platform specific model from which source code can be generated. Using an abstract model to define the complexity of dependencies found in deployment allows for transforming that model to deployment artifacts that ensure the correct configuration and setup of a software system for a specific runtime environment. This integrates with the idea of a computation-independent, platform-independent and platform-specific model taken from MDA [MM04]. Deployment scenarios, as the problem domain, are captured in computation-independent models. The platform independent model represents the deployment configuration for a particular environment. The final platform specific model already reflects the characteristics of a runtime environment, such as a JBoss application server in a productive setting.

Consequently, by applying the idea of models at different levels of abstractions, our approach enables a separation of responsibilities reflected in software development roles. While defining a deployment model at a high level of abstraction is the task of a software architect, the definition of transformation rules and templates for the platform specific model is the responsibility of a deployment manager. Furthermore, during application and change management this will enable higher efficiency and productivity of the change management processes. Considering the choice of language for model definition we see two possible options: Using a general purpose language and refining it for a specific domain problem or using a domain specific language. In the domain problem of deployment, we see UML as a special case because UML as a general purpose language already offers the ability to model deployment scenarios as part of the language.

For a prototype implementation we will use UML [UML04] profiles while the choice of a domain specific language will be left open in further phases. We will use the ability of UML for creating profiles allowing for the refinement of meta models that come with the language. In our specific case we will create UML profiles for deployment models that can be applied for a specific deployment scenario. Furthermore a profile allows for a model to be defined for a specific runtime environment such as a particular application server. As shown in Figure 1(a), the version information is included in the artifact model using a special `Version` tag. This allows representations of dependancies of versioned artifacts as shown in Figure 1(b). In addition, the stereotype `ejb-jar` is defined in a J2EE-Profile. The application of multiple profiles allows for the representation of independent concerns. In order to define constraints in the resulting model the profile offers the use of OCL[OCL06] constraints on stereotypes.

By means of transformation rules and templates, the model is transformed into concrete tasks that are executed for deployment. We envision the use of AndroMDA [And] and



(a) Part of UML 2.0 Profile (b) Deployment diagram with profiles applied

Figure 1: Profile Model and Application

Apache Ant files for the transformation. Executing the final Ant files will result in assembling the application components in their respective versions and corresponding artifacts where all configuration and version information has been derived from the deployment model.

3 Outlook

In this paper we outlined a model-driven approach for deployment management. We propose the use of models adaptable to the required level of detail and capable of capturing the interdependencies of components, artifacts and their versions.

As a first step, we will define a UML profile for deployment models that captures generic requirements found in deployment scenarios such as compontents, artifacts and versions. With UML already offering deployment diagrams per definition, a respective profile will extend the basic notions considering advanced deployment scenarios. After the specification of a meta-model, transformation rules in the form of an AndroMDA cartridge and respective templates will be developed. This setup will be used to provide the proof-of-concept of our approach. After successful proof of concept we will consider to change the model language from a general purpose language to a domain specific language. We believe that UML is applicable for showing the practicability of our approach. However, for the complexity of large enterprise applications we envision the use of a domain specific language to be based on open standards such as the Eclipse Modeling Framework [EMF07]. The choice of EMF will also enable the use of GMF [GMF] which will ease the development of required user interfaces.

A remaining challenge in our solution will be the versioning of the model itself. A particular model instance is seen as a deployment configuration version that is a snapshot and will be revised over time. The versioning of a model is needed to satisfy the requirements of application and change management where a new release of an application leads to a new version of the corresponding deployment model. The challenge of model versioning has been addressed in [YLG04] and will be the topic of [UML07] for proposing solutions for managing the evolutionary characteristics of models.

References

[And] AndroMDA. Available at http://www.andromda.org/ (last checked: 2007-01-28).

[Ant] The Apache Ant Project. Available at http://ant.apache.org/ (last checked: 2007-01-28).

[Che76] Peter Pin-Shan Chen. The entity-relationship model toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976.

[DMG07] Paul Duvall, Stephen M. Matyas, and Andrew Glover. *Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series)*. Addison-Wesley Professional, 2007.

[Ecl] Eclipse - an open development platform. Available at http://www.eclipse.org, (last checked: 2007-01-28).

[EMF07] Eclipse Modeling Framework. Available at http://www.eclipse.org/emf/ (last checked: 2007-01-28), 2007.

[GMF] Eclipse Graphical Modeling Framework. Available at http://www.eclipse.org/gmf/ (last checked: 2007-01-28).

[Her03] Helmut Herold. *Das Profitool zur automatischen Generierung von Programmen*. Addison-Wesley, 2003.

[JSR06] JSR 244: JavaTM Platform, Enterprise Edition 5 Specification. Available at http://jcp.org/en/jsr/detail?id=244 (last checked: 2007-01-28), 2006.

[JWS06] Java Web Start Technology. Available at http://jcp.org/en/jsr/detail?id=56 (last checked: 2007-01-28), 2006.

[Mav] Apache Maven Project. Available at http://maven.apache.org/ (last checked: 2007-01-28).

[MKS] MKS Integrity. Available at http://www.mks.com/products/ (last checked: 2007-01-28).

[ML05] Jeff McAffer and Jean-Michel Lemieux. *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java(TM) Applications*. Addison-Wesley Professional, 2005.

[MM04] Joaquin Miller and Jishnu Mukerji. Model Driven Architecture. Available at http://www.omg.org/docs/ormsc/01-07-01.pdf (last checked: 2007-01-28), 2004.

[OCL06] Object Constraint Language, OCL 2.0. http://www.omg.org/docs/ptc/03-10-14.pdf (last checked: 2007-02-21), October 2006.

[UML04] Object Management Group: UML 2.0 Specification. Available at http://www.uml.org/, (last checked: 2007-01-28), Oktober 2004.

[UML07] Workshop "Vergleich und Versionierung von UML-Modellen" co-located with "Software Engineering 2007" Hamburg, Germany, March 2007.

[YLG04] Jing Zhang Yuehua Lin and Jeff Gray. Model Comparison: A Key Challenge for Transformation Testing and Version Control in Model Driven Software Development. In *In Object Oriented Programing, Systems, Languages and Applications*, 2004.