Grid Development Tools for Eclipse

(Position Paper)

Thomas Friese, Matthew Smith, Bernd Freisleben

Department of Mathematics and Computer Science, University of Marburg
Hans-Meerwein-Str., D-35032 Marburg, Germany
{friese,matthew,freisleb}@informatik.uni-marburg.de

1 Introduction

Grid Computing [1, 2] has gained a huge momentum not only in the scientific community but also in industrial application areas. The technology itself has evolved from proprietary, specialized applications and middleware to the adoption of web service standards [3, 4]. Service-oriented Grid environments offer the possibility to request and use computing power, data access, network bandwith and special devices on demand, ultimately leading to dynamically establishing even the Grid infrastructure itself on demand in service-oriented ad hoc Grid computing [5]. A crucial problem is lowering the barrier to develop Grid applications running in such environments. Helping non-Grid experts to translate their domain knowledge into Grid enabled applications is a key factor in enabling more parties to collaborate in problem solving and resource sharing and form a vibrant global Grid ecosystem.

To provide such support, both the *runtime* complexity of Grid systems and the *development* complexity of distributed Grid applications must be reduced. Our *Grid Development Toolkit for Eclipse* (GDT) [6] project focuses on mastering the complexity of the application development process for service-oriented Grid environments. Most modern service-oriented Grid middleware systems are developed using the Java language, making the Eclipse platform an ideal application development environment. The GDT project aims at extending the Eclipse platform with development tools allowing the easy creation and debugging of Grid services and clients. In addition to supporting developers in pure Grid service based application development, additional management functionality for Grid environments is provided by the GDT.

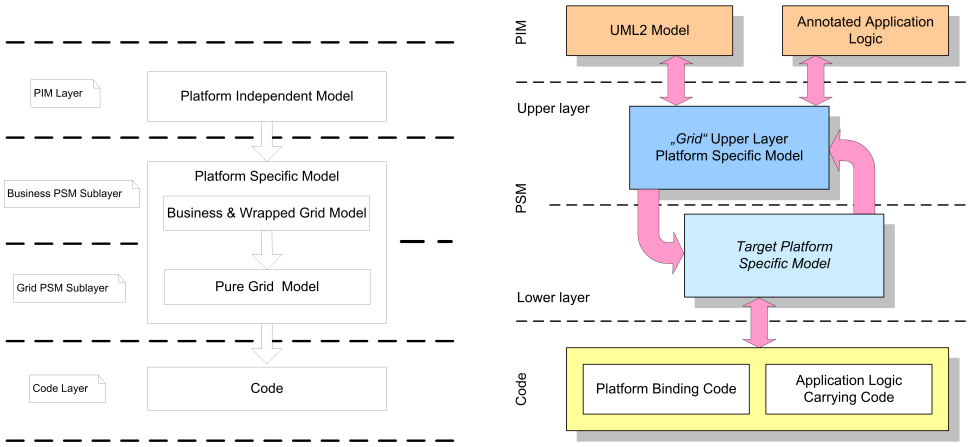
2 GDT Features

The GDT project follows a model-driven development approach [7] both for its internal structure as well as the intended tool functionality. In previous work [8], we have proposed to divide the Platform Specific Model (PSM) layer of the model-driven development approach for the Grid into two parts, as shown in figure 1(a). This separation into upper and lower layer PSM enables domain experts to concentrate on the application logic and keep their domain specific model apart from the target system specific model. A target system specific model for a concrete *target platform* (i.e. Grid middleware implementation such as the Globus Toolkit 4 [9] or Unicore-GS [10]) can then be derived from the upper layer Grid specific application model. On this lower layer, the domain experts for a concrete Grid implementation may modify the model and implementation and concentrate on implementation specific issues.

Figure 1(b) shows the flow of model information between the actual Grid service implementation, the upper and lower layer PSM and the different model sources usable as input. As a primary source of model information, the GDT currently uses Java annotations in the classes representing the core (business) logic of a Grid service (the *annotated service source*). The GDT also provides wizards to create skeletons for this annotated service source that a developer can fill in and extend with custom logic. We are currently developing a module allowing for model import from an UML editor using a profile for service-oriented Grid applications. As a last source of model information, the target platform mapping model should enable the GDT to read service model information from the service implementation allowing for analysis and import of existing service implementations.

The mapping between upper and lower layer PSM models is automated in the GDT, leaving developers with the option to transform their applications into Grid services by simply tagging them with Java annotations. In contrast to pure template and wizard based solutions, the internal model-to-model transformation offers the ability to keep dependent parts of the resulting source code synchronized even if the user chooses to manually edit the generated platform bindings (a common case on e.g. the GT4 platform).

The ability to use an interactive debugger during the development process is another strong aid in the development process of a Grid based application especially since the involved application containers and tools tend to reach a high level of complexity. Therefore, the GDT offers components that gather information about the



(a) Grid revised model stack of the MDA.

(b) Flow of model information within the GDT.

Fig. 1.

deployed components and their versions in remote Grid service containers and assists the developer in attaching the Java debugger to the remote container and debug the live service.

Grid applications are often large scale complex applications that also require a substantial amount of maintenance during deployment and use. We therefore are currently integrating Grid monitoring and discovery tools as well as a graphical representation of the Grid environment into the Eclipse workbench in order to monitor the active Grid resources and allow for integrated development and management of the application. Those management components are also reusable as Rich Client Platform applications geared towards Grid system administrators instead of application developers.

In addition to the interactive development tools, it is desirable to use the target platform mapping model in an automated build or testing environment. The GDT therefore provides headless application support as well as a custom Ant task, allowing users to annotate service logic carrying code and transform it into a deployable Grid service with a single command. We use the Ant support together with Eclipse release engineering technology for automated testing of the GDT functionality against a set of unit tests checking the GDT for coverage of the Grid service standards.

3 Use and Integration of Eclipse Technology

The GDT uses the Eclipse Modeling Framework for internal representations of the upper and lower layer PSM. Java Emitter Templates (JET) are used to generate source code and deployment artifacts from the models. JMerge is used to transfer user modifications in the source code into newly generated sources. The GDT uses JDT annotation processing support (APT) and the Java DOM representation to collect information from Java compilation units, and it provides custom builders that perform the various transformations based on resource changes.

4 Issues with Eclipse

In the current implementation of the GDT, Java Emitter Templates are used to generate source code from application models. Due to the template syntax and its representation of the structure of the resulting source code, these templates proved hard to be managed with increasing model complexity. Furthermore, the generation of artifacts is rather coarse grained, which raises concerns about performance issues in the future and with more complex internal dependencies. Many Grid specific additions to web service standards and custom XML based formats made it hard to use the models defined in the Web Tools Project [11]. We found that following the changing structure and implementation of a large number of Eclipse projects increases the required development overhead significantly.

5 Related Work

The Web Tools Project (WTP) [11] provides wizards for the creation of Java Web Services and models for the representation of standard web service artifacts. These tools do not deal with Grid specific issues at the moment, but are geared towards tight AXIS (a Java Web Service Framework) integration. Furthermore, *round trip support*, i.e. allowing developers to change the model or the generated sources and reflect the changes in all corresponding model or source elements is currently not supported by WTP. However, the WTP offers many useful editors for web service related formats such as WSDL files that can be seamlessly used for artifacts created and managed by the GDT.

The Generative Model Transformer (GMT) [12] project's aim is to "produce a set of prototypes in the area of Model Driven Engineering". ATL is an emerging subproject in the GMT project aimed at providing development tools for the Atlas Transformation Language [13]. We are currently investigating the possibility to specify the model-to-model transformation from upper layer to lower PSM using the Atlas Transformation Language.

References

1. Berman, F., Fox, G., Hey, T.: Grid Computing: Making the Global Infrastructure a Reality. Wiley (2003)
2. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (2003)
3. Foster, I., Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Kishimoto, H., Maciel, F., Savvy, A., Siebenlist, F., Subramaniam, R., Treadwell, J., von Reich, J.: The Open Grid Services Architecture, Version 1.0 (2004) `https://forge.gridforum.org/projects/ogsa-wg/document/draft-ggf-ogsa-spec/en/`.
4. OASIS: Web Services Resource Framework (2004) `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf`.
5. Smith, M., Friese, T., Freisleben, B.: Towards a Service-Oriented Ad Hoc Grid. In: Proceedings of the 3rd International Symposium on Parallel and Distributed Computing, Cork, Ireland, IEEE Press (2004) 201–209
6. The Distributed Systems Group, University of Marburg, Germany: (Grid Development Tools for Eclipse) `http://ds.informatik.uni-marburg.de/MAGE/gdt`.
7. Object Management Group, Inc.: MDA Guide 1.0.1, omg/2003-06-01 (2003)
8. M. Smith, T. Friese, B.F.: Model Driven Development of Service-Oriented Grid Applications. In: Proc. of the International Conference on Internet and Web Applications and Services, Guadeloupe, IEEE Press (2006)
9. The Globus Alliance: (Globus Toolkit 4.0) `http://www.globus.org`.
10. UNICORE: (Uniform interface to computing resources) `http://unicore.sourceforge.net`.
11. The Eclipse Project: (Web Tools Project) `http://www.eclipse.org/webtools`.
12. Eclipse.org: (Generative model transformer) http://www.eclipse.org/gmt.
13. Jouault, F., Kurtev, I.: Transforming models with atl. In: Proceedings of the Model Transformations in Practice Workshop at MoDELS 2005, Montego Bay, Jamaica (2005)