Open Source Tools for Software Product Line Development *

Sergio Segura, David Benavides, Antonio Ruiz-Cortés and Pablo Trinidad
Department of Computer Languages and Systems
University of Seville
email:{segura, benavides, aruiz, trinidad}@tdg.lsi.us.es

Abstract

Open-Source (OS) software development differs widely from close-source development practices because of a number of reasons: project organization, distributed developers, code-centric, etc. These characteristics force the development tools used in the context of OS development to fulfill a set of requirements such as being extensible, multi-platform, version control support, etc. In this paper we set the basis for a discussion about the features that a success OS tool for the development of Software Product Lines (SPLs) should provide. As a starting point, we analyse the projects of four major OS development tool and summarize its main features in a reference feature model. Next, we introduce some the most popular OS feature modeling tools available in the SPL community and check how they support the identified features.

1 Introduction

The combination of SPL engineering and OS software development is expected to become a profitable strategy for intra-organizational reuse improving product quality, costs and time-to-market [5]. In this context, providing the OS community with efficient SPL development tools emerges as an inevitable step for powering the promising relation between both fields.

In [8] we presented FAMA, a framework for the visual edition and automated analysis of feature models. Currently, we are considering releasing it as an OS project. However, we found that running a success OS project is a hard task which motivates multiples questions:

- What features should be supported by a success development tool?, i.e. multi-platform, extensible, etc.
- What kind of enabling tool should be provided? Are the services provided by OS repositories enough?
- How powering the communication between developers and users? Are mailing lists and wikis the only suitable mechanisms?
- What kind of documentation should be provided? Which formats are used in the OS community?
- Do SPL development practices require any specific tool support?

Detailed instructions about how running an OS project are well documented in the literature [10]. However, not all such instructions are always followed in practice. In a similar way, the support tools and features provided by the different OS projects usually differ from each other.

In this paper we try to answer previous questions setting the basis for a discussion about the features that a success OS tool for the development of SPLs should provide. As a starting point, we analyse the project of four success OS development tools and summarize its main features in a reference feature model. Next, we analyse some of the intrinsic characteristics of SPL that should be considered when designing an efficient OS tool for SPL development. Finally, we introduce some of the most popular OS feature modeling tools available in the SPL community and check how they support the identified features.

The remainder of this paper is organized as follow: in Section 2, OS development tools are analyzed and a feature model summarizing its main features is presented. Specific requirements of SPL development tools are studied in Section 3. Section 4 overviews the features supported by some popular OS feature modeling tools. Finally, we summarize our main conclusions in Section 5.

2 Open Source Development Tools

In this section we explore the general features that a success OS development tool project should provide. In particular, we focus our analysis on three directions:

*This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project Web–Factories (TIN2006-00472)

1. Features supported by the tool. Hence, for instance, a success OS development tool should be extensible, multi-language, updatable, etc.

2. Set of enabling tools and technologies that make possible the development and collaboration between the developers of the project. This is a crucial aspect in OS development since developer will not contribute to the project if they do not have efficient mechanisms to do it. Some examples of these kinds of tools and technologies are Concurrent Version Systems (CVS) or Wikis.

3. Documentation provided by the project and formats in which it is published, i.e. Frequently Asked Questions (FAQ), Tutorials, etc.

Notice that although we focus our analysis on the relevant features that a success OS development tools should support, such features are general enough to be applicable for any kind of OS project.

In order to collect the information about the required features we propose analyzing the projects of four major OS development tools with solid reputation in the software industry. In particular, we propose studying a representative set of different kind of development tools composed by two general purpose IDEs (Eclipse, Netbeans) a Web development framework (Apache Struts) and a specific purpose IDE (ArgoUML). The main goal is to obtain a reference feature model with the features that a success OS development tool project should support. In the next sections, the analyzed OS projects are presented.

2.1 Eclipse

Eclipse [4] is a Java-based, extensible, development framework. It is based on a core plug-in architecture and a set of services for building Integrated Development Environments (IDE). Hence, with a suitable set of plug-ins, Eclipse can be used as a powerful cross-platform IDE for multiple kind of development such as programming, visual modeling, or text edition. The Eclipse Project was originally developed by IBM back in 2001. Today, the project is leaded by the Eclipse Foundation, a non-for-profit corporation of software industry vendors which manages and directs Eclipse's ongoing development.

The Eclipse Foundation provides a complete infrastructure for the development and usage of Eclipse. Control access to the source code is managed using a CVS code repository. Bugs and open issues are reported and managed using a set of Bugzilla databases. All the information about the project is published in the official Web site including abundant documentation for users and developers such as tutorials or FAQs.

Eclipse can be easily extended and updated from the Internet using a wizard integrated in the tool. In a similar way, documentation and multi-language support are also available as a part of the framework.

The collaboration and communication between developers is also supported by the Eclipse Foundation by means of several mailing lists, newsgroups and an official Wiki.

2.2 Netbeans

The *NetBeans IDE* [6] is a multi-platform IDE for software developers. All the functions of the IDE are provided by modules. Each module provides a well defined function, such as support for the Java language, editing, or support for the CVS versioning system. Modules also allow NetBeans to be extended with new features such as support for other programming languages or visual modeling. The NetBeans IDE was released as an OS project in June 2000 by Sun which still today continues acting as the steward of the project.

Netbeans is available in multiple languages and can be easily downloaded and updated from the Internet. Its source code is available within a CVS code repository which can also be examined from the Web Site. Bugs are reported and managed using an issue tracking system. The documentation for users and developers in mainly composed by tutorials and FAQs. Finally, communication between developers and users is mainly powered by several mailing lists and wikis.

2.3 Apache Struts

Apache Struts [1] is a Web application framework for developing J2EE Web applications. It provides a set of extensible components to encourage developers to adopt a Model-View-Controller (MVC) architecture. Although it is distributed as a jar package, there exist multiples extensions for well known IDEs which make use of it. Struts was donated to the Apache Foundation in 2000 becoming a top level project of the Apache Software Foundation in 2005.

The Apache Software Foundation provides multiple mechanisms to attract contributions to the projects. Such mechanism includes a SVN code repository, an issue tracker, an official Web site for publishing information and multiple download sites. Documentation is mainly published by mains of tutorials and FAQs. Additionally, developers and users can communicate each other's using any of the multiple mailing lists, wikis or the issue tracker.

2.4 ArgoUML

ArgoUML [2] is an UML modeling tool. It runs on any Java platform and is available in multiple languages. ArgoUML was born in the academic field and was released

as an OS project in 1999. Since then, it has evolved and has been used as the base for other modeling tools such as Poseidon for UML[1] or MyEclipse[2].

ArgoUML code is stored in a SVN repository. Issues and defects are managed using an issue tracker. The tool can be downloaded from the Web site of the project which also give access to abundant documentation in form of FAQs, tutorials and a cookbook. The collaboration between users and developers of ArgoUML is mainly carried out by means of mailing lists and the issue tracker.

2.5 A Reference Feature Model

Figure 1 depicts a feature model summarizing the features identified in the studied OS projects. The features detected in all the projects are modeled as mandatory meanwhile the features identified only in some of them are defined as optional.

According to the model, success OS development tools projects should provide, at least:

- A set of enabling tools for supporting the development and diffusion of the project. In particular, OS projects should include a dedicated Web site, a control version system and a bug/issue tracker.
- Support for specific features such as being extensible, updatable, multi–platform and multi–language. Additionally, it is also desirable to provide built–in support for control version as a part of the development tool.
- Documentation in different formats for both, users and developers. Common formats are FAQs/How–to, tutorials and cookbooks.
- Mechanisms to power the communication and collaboration between users and developers. Frequent mains for such purpose are mailing lists, wikis and newsgroups.

3 Software Product Line Development Tools

SPL development practices are frequently very different from the one used when developing individual software products. Therefore, such characteristics should be taken into consideration when analyzing the features that a suitable OS tool for SPL development should support. In [11], Gurp gives a step forward in this context by exposing two major problems detected in SPL tools when comparing it with OS tools:

- *Lack of integration*. In contrast with the OS environment in which tools usually integrate into other tools, SPL development tools tend to be stand–alone. This force to use different tools on the different stages of SPL development making the synchronization between models, source code and documentation a time–consuming and error–prone activity. Additionally, the lack of integration commonly lead to interoperability problems between different development tools.
- *Specific purpose*. SPL development tools are not usually general purpose. By contrast, OS developer tend to use general purpose tools in which they find all they need.

In order to overcome the problems pointed by Gurp we suggest that OS SPL development tools should be integrated into other general purpose tool widely extended in the OS community such Eclipse or Netbeans. The advantages of doing this are manifold:

- SPL developers can use a single tool in all the stages of development overcoming the integration and interoperability problems derived from using different development tools.
- It is not necessary to start the development of the tool from scratch. Hence, for instance, the Eclipse platform provides all the frameworks and services needed for building complete IDEs as plug–ins of the Eclipse IDE. Furthermore, plug-ins deployed in Eclipse are automatically multi–platform, updatable and partially extensible.
- The barrier of entry for new contributors is lower since they are probably familiar with the tool.

4 OS Software Product Line Development Tools

In this section we overview some of the OS tools available in the SPL community and check how they support the identified features. In particular, we introduce three popular OS feature modeling tools and check if they support the features presented in Figure 1. Additionally, we check if such tool are integrated into other general purpose tools avoiding the problems described in Section 3.

Feature Model Plug-in (FMP) [7] is implemented as an OS Eclipse plug–in. The project is hosted in an OS repository which provide it with a standard set of support tools such as a CVS, mailing lists or a bug tracker. Additionally, the project provides a dedicated Web site[3] with information about the tool and documentation in different formats. FMP

[1] www.gentleware.com/

[2] http://www.myeclipseide.com/

[3] http://gp.uwaterloo.ca/fmp/

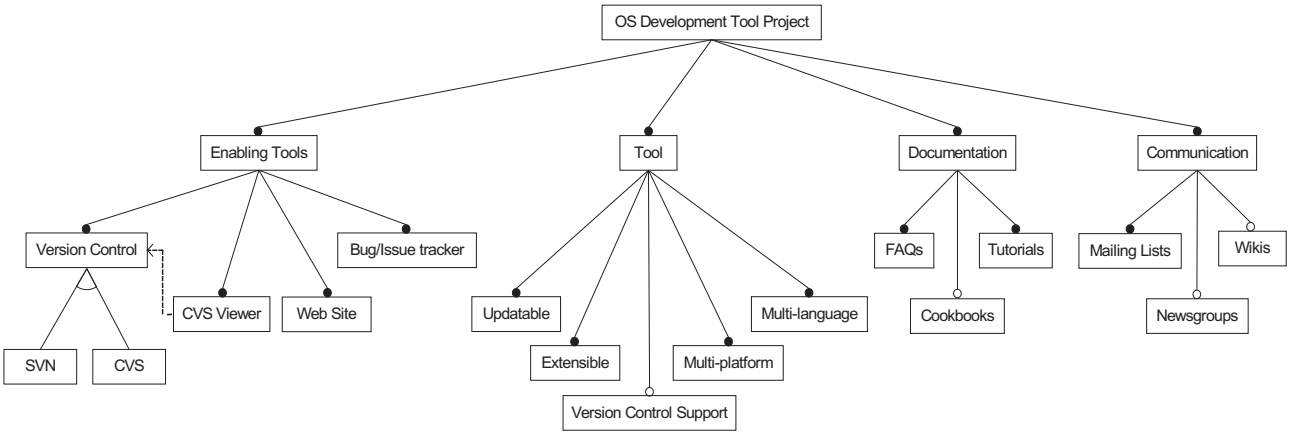

Figure 1. OS Development Tool Project

is distributed directly as a downloadable plug-in making the automated updating of the tool not possible. In addition, specific extension mechanisms or tutorial for developers are not available.

XFeature [9] is an XML-based feature modeling tool implemented as an extensible and updatable Eclipse plug–in. Although the project is recognized as an open source effort, it does not provide mechanism to support it such as a CVS or mailing lists. Documentation in different formats can be found on the Web site of the tool[4].

CaptainFeature [3] is a feature modeling tool implemented as a stand-alone Java application. It cannot be updated automatically. Similarly, it does not provide extension mechanisms or development tutorials. The project is hosted in a OS repository which provides it with common support tools. The documentation of the tool is represented by a simple user guide accessible from the help menu in the tool.

Table 1 summarizes the features supported by the presented tools.

5 Conclusions

In this paper we set the basis for a discussion about the features that a success OS tools for SPL development should provide. In particular, we first study four major OS development tools and summarize its main features in a reference feature model. Next, we argue about the intrinsic characteristics of SPL development and suggest that SPL development tools should be integrated into other general purpose tools. Finally, we overview some popular OS feature modeling tools and check how they support the identified features.

[4]http://www.pnp-software.com/XFeature/

The work reveals that none of the studied tools provide all the features identified as mandatory. Additionaly, it also shows that although OS repositories offer a good basic support they do not provide all a success OS project needs.

Although our approach is not the result of a rigorous analysis, we consider it clarify some of the multiple aspects that should be considered when running an OS project. Hence, if we finally decide to release FAMA as an OS tool, we will definitely work to provide the identified features.

References

[1] Apache Struts. http://struts.apache.org/.

[2] ArgoUML. http://argouml.tigris.org/.

[3] CaptainFeature. https://sourceforge.net/projects/ captainfeature/.

[4] Eclipse. http://www.eclipse.org/.

[5] ITEA COSI Project. http://www.itea-cosi.org.

[6] Netbeans. http://www.netbeans.org/.

[7] M. Antkiewicz and K. Czarnecki. FeaturePlugin: feature modeling plug-in for Eclipse. In *eclipse '04: Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange*, pages 67–72, New York, NY, USA, 2004. ACM Press.

[8] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. FAMA: Tooling a framework for the automated analysis of feature models. In *Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS)*, pages 129–134, 2007.

	FMP	XFeature	CaptainFeature
	Tool		
Extensible	+	+	-
Updatable	-	+	-
Multi-language	-	-	-
Multi-platform	+	+	+
Version Control Support	-	-	-
Integrated into other tool	+	+	-
	Enabling Tools		
Dedicated Web Site	+	+	-
Control Version System	+	-	+
C.V. Web Interface	+	-	+
Bug/Issue Tracker	+	+	+
	Documentation		
FAQ / HOW-TO	+	+	-
Tutorials	+	+	+
Cookbooks	-	-	-
	Communication		
Mailing Lists	+	-	-
Wikis	-	-	-
Newsgroups	-	-	-

Table 1. OS Feature Modelling Tools

[9] V. Cechticky, A.Pasetti, O. Rohlik, and W. Schaufelberger. Xml-based feature modelling. *LNCS, Software Reuse: Methods, Techniques and Tools: 8th ICSR 2004. Proceedings*, 3107:101–114, 2004.

[10] K. Fogel. *Producing Open Source Software: How to Run a Successful Free Software Project*. O'Reilly Media, Inc., 2005.

[11] Jilles van Gurp. OSS Product Family Engineering. In *First International Workshop on Open Source Software and Product Lines (OSSPL)*, Baltimore, Maryland, USA, August 2006.