Teaching Eclipse Plug-in Development for Undergraduates

Tianchao Li and Michael Gerndt
Institut für Informatik,
Technische Universität München
{lit,gerndt}@in.tum.de

ABSTRACT

With the rapid adoption of Eclipse as both development tool and application platform, teaching Eclipse plug-in development in universities is showing its necessity and advantage, especially on the undergraduate level. This paper introduces our practice in teaching Eclipse plug-in development for undergraduate students in Technische Universität München, Germany. Following a set of carefully designed lab courses, the students understand the mechanism of Eclipse from ground up - starting from the underlying SWT/JFace toolkit and the extension mechanism of the Eclipse runtime up to the Eclispe Rich Client Platform (RCP) and Integrated Development Environment (IDE). Students work on mini projects to have a practical experience in designing and developing Eclipse plug-ins. We summarize our experiences in the preparation of background knowledge and designing exercises, and make suggestions for improving Eclipse from a teaching and learning point of view.

1. INTRODUCTION

Eclipse [2] has been rapidly adopted in companies, academia and education as a development tool. The open source platform, well-defined extension mechanism, abundance of plug-ins and good documentation have all made this happen.

On the other hand, teaching Eclipse plug-in programming at universities is still not a popular practice nowadays. The complexity of the Eclipse platform itself is one of the reasons - SWT/JFace, extension mechanisms, and the vast extension points, each is itself an issue. Teaching Eclipse plug-in programming for undergraduate students is even more rare and challenging. In addition to the complexity of Eclipse itself, Eclipse plug-in development also has a lot of prerequisite knowledge in object-oriented programming, software engineering (esp. UML and design patterns) and XML.

However, teaching Eclipse plug-in development for undergraduate students is both necessary and helpful. With the rapid adoption of Eclipse, the mastering of Eclipse plug-in development can both enhance the competency of graduates in object-oriented programming and create a basis of knowledgeable programmers for deploying Eclipse in various research projects. Teaching Eclipse plug-in development will also make Eclipse even more popular both as a development tool and an application platform.

At Technische Universität München we are offering such a course on programming Eclipse plug-ins for undergraduate students.

The Eclipse lab course has been offered to a small group of students in the summer semester of 2005 for the first time. The attendees are students from undergraduate studies, mostly in their 3rd semester. They have at least some basic knowledge about object-oriented programming in Java and XML before they take the lab course. In the past winter semester of 2005/2006, this course was offered a second time, however for the most talented students in their first year of study in computer science, within the CS department's elite program.

The attendees are those with very good performance in the high school graduation exam and/or excellent computer background, who will probably be the best students when they graduate. These students are offered special courses and exercise groups. The goals are manyfold. First, those student should get into contact in their early semesters so that they can benefit from those contacts in the later semesters. Second, we want the students to get a broader overview and understanding of computer science very early. They can master this additional information based on their excellent background and will be able to combine this knowledge with the more theoretical base courses taught in the first years.

Given their excellence, teaching Eclipse plug-in development for such freshmen that are not assumed to have much knowledge in programming is especially challenging. As a result, the content of the course has been adjusted and extended a little bit to give more background knowledge and preparations that are essential for the students to start programming Eclipse.

This paper introduces our experience in teaching Eclipse plug-in development for undergraduates. The remainder of this paper is organized as follows: Section 2 introduces the general structure of the course, which is divided into lectures, exercises, and mini projects. Section 3 provides detailed design of the hands-on exercises, and discusses some relevant issues concerning Eclipse itself. Section 4 presents the miscellanies, including the preparation of necessary background knowledges, online course materials, teaching techniques etc. The paper concludes with a short summary in Section 5.

2. GENERAL COURSE DESIGN

ECOOP'06 Eclipse Technology eXchange (ETX) Workshop, July 4th, 2006, Nantes, France

Many existing books and materials about Eclipse plug-in programming follow the same pattern - each chapter focus on a different set of relevant extension points of Eclipse and guide the reader with examples on defining extensions to the specific extension points. This works well for people with good knowledge of programming, but is however not suitable for undergraduate students in general.

The length of our course is 3 hours per week and a typical semester has 14 weeks at German universities. Mastering Eclipse plug-in programming from the level of freshmen within such a short course is quite demanding. The structure of our course has to be adapted to this fact and tries to guide the students with a flat and steady learning curve. Generally speaking, the course is divided into three parts: lectures, exercises, and mini projects.

In the lectures part, object-oriented programming and UML are introduced first. These are the background knowledge that will be very helpful for the student to start with Eclipse programming and to understand the general picture of extensions. In addition, Eclipse is also introduced, with an emphasis on its general architecture and extension mechanism.

A sequence of exercises constitute the major part of the lab course, which offer hands-on training of different aspects and different levels of knowledge for Eclipse plug-in development. Following the detailed instructions on the exercise sheets, students work individually on the assignments. The exercises are designed in such a way that the student master the relevant knowledge from ground up - starting from programming with SWT, JFace, followed by understanding extensions and extension points, defining applications for the Rich Client Platform (RCP), and to extend the complete Integrated Development Environment (IDE). Because of the vastness of different extension points in the Eclipse platform and the limited timeframe in the course, it is impossible to cover every aspect of Eclipse extension. Thus, the emphasis in the exercises is to guide the students with a series of assignments to build up all the necessary knowledge and practice in defining extensions to the Eclipse platform. For extensions to specific extension point provided by the Eclipse platform, the students have enough knowledge on how to utilize the documentation to get further details and to apply their general knowledge about extensions to actually implement the extension.

A complete training on Eclipse plug-in development requires the ability to design a specific extension and to organize a set of extensions. This cannot be achieved without actual development experience. In the last session of the lab course, some mini projects are distributed to groups of students. The number of students of each group is quite flexible (typically 2 to 3 students) and the complexity of the exercises can be adjusted according to their actual level. Utilizing their knowledge mastered in the hands-on exercises, they discover the necessary extensions, seeking for appropriate extension points and even explore the Eclipse source code for necessary information. Regarding the level of knowledge, certain students can be offered even more advanced topics like programming graphical editors with the Eclipse Modeling Framework (EMF) and Graphical Editing Framework (GEF), etc.

3. HANDS-ON EXERCISES

The hands-on exercises constitute the most important part of the lab course. The first part of the exercises focus on programming standalone Java applications, especially those GUI applications utilizing the SWT/JFace libraries. The second part of the exercises focuses on the extension mechanism of the Eclipse platform and the headless, RCP, and IDE applications utilizing that mechanism. The mini projects, though not part of the hands-on exercises, are actually a continuation of the exercises that teach the students the development of extensions.

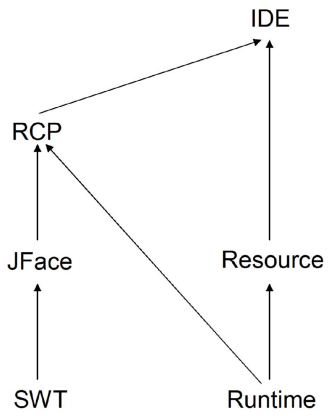

Figure 1: Course Roadmap

The exercises cover the most fundamental parts of Eclipse (see Figure 1 for an exercise road-map). The more advanced and also optional components of the Eclipse platform, e.g., the search, debug, and update componentes etc, are not covered.

3.1 Java Programming in Eclipse

One prerequisite of Eclipse plug-in development is the development of Java applications with Eclipse and the usage of the Eclipse specific GUI libraries SWT and JFace. The sequence of exercises in this part includes:

Lab 1: Java Development Tool (JDT) This leads the students into the world of Eclipse by doing Java programming with JDT. The familarity with JDT is not only important for the students to program stand-alone SWT and JFace programs in the following exercises of this part but also critical for the students to continue with development of plug-ins with the Plug-in Development Environment (PDE) in the next part.

The scope of the exercise covers the most important features of the Java editing facilities like code assist and refactoring etc., and the setting of build and runtime class paths and runtime libraries. This prepares for the following exercises in SWT and JFace, especially for earler versions of Eclipse. In our experience, students are usually confused of the configurations necessary for creating stand-alone SWT and JFace applications. Eclipse 3.1 has improved support for configuring and launching stand-alone SWT/JFace applications, which are very welcomed by our students.

Since the course is given for the first semester students, an introduction to object-oriented programming is given. In this lab course the students experiment with the Java editing facilities of Eclipse while working on examples to extend their Java knowledge.

Lab 2: Standard Widget Toolkit (SWT) As the fundamental UI toolkit, SWT provides the basic building blocks of Eclipse and graphical Eclipse plug-ins. This lab leads the students to program stand-alone Java programs with SWT, covering the different SWT controls, layouts and event mechanisms.

To be frank, learning Eclipse plug-in development does not necessarily require the students to know about programming stand-alone applications in SWT. However, we feel it appropriate for our course as we believe to understand the mechanism is more important than only being able to program the following examples.

Lab 3, 4: JFace JFace complements SWT with higher level application support. This is a relatively large topic, as it contains several different smaller frameworks like viewers, contributions and actions, dialogs and wizards, registries for images and fonts, as well as a text editing framework. Two labs go for programming stand-alone applications using JFace (together with SWT, of course), covering some of the most commonly used techniques - application window and dialogs, viewers and image handling, as well as actions, menu and toolbar support. The text editing framework, as a separate topic, is not covered. The wizards, as a common topic, are to be covered in the plug-in labs.

Having introduced programming stand-alone applications using SWT, JFace becomes the natural step forward. We believe that students attending our course should learn something that can be used in different situations and will be very helpful in their future career. Currently, JFace depends on Eclipse core (commands and runtime), which in turn depends on OSGi. In order to program a stand-alone application with JFace, a lot other dependent libraries have to be included in the class path. An inspection of JFace source code reveals that the dependencies are only for status, progress monitor, job/operation/runnable, and adaptation support. We feel it better if the Eclipse core is refactored into an extension mechanism relevant part (Extension Runtime) and irrelevant part (Common Runtime), as illustrated in Figure 3.1. This follows the current approach of having separate *org.eclipse.text* and *org.eclipse.core.commands* plug-ins from the core runtime.

Lab 5: GUI Programming Using Visual Editor Programming can be tedious work if all the GUI design has to be done by ”hand coding”, which usually involves significant amount of time dealing with the details of various layouts, constraints, and settings. Before we transit to the next part to work on plug-ins, we want the students to be skilled in programming GUI elements with all the help they can get - the Visual Editor (VE) for visually programming visual classes is one such tool.

We choose not to use the VE when starting to teach GUI programming, because we believe that the students need some time to understand the concepts of GUI coding. Also, this part is actually a new addition in the next semester's teaching, mainly because of the recent progress in the functionality and usability of VE, noticeably the new GridLayout tooling and the improved support of parsing hand-written code. We can expect more wider adoption of this tool than before.

3.2 Extending Eclipse Platform

For the development of Eclipse plug-ins, we introduce every aspect of it. The sequence of lab courses in the second part includes:

Lab 6, 7: Eclipse Runtime and Extension Mechanism This introduces Eclipse runtime, with a focus on its extension mechanism. Topics covered includes both the extension of an existing extension point and the definition of a new extension point. The exercises start from the implementation, configuration, and launching of a headless application that extends the *org.eclipse.core.runtime.applications* extension point and outputs greeting messages. Then, students work on the definition of an extension point that defines the name of the person to be greeted and on the customized class contribution, which is to be provided by individual extension definitions. We try to eliminate GUI topics from this lab so that the students can focus only on the extension mechanism provided by the Eclipse runtime.

As the next step, we designed an exercise for the students to implement an extensible GUI based on the SWT, JFace, and the Eclipse runtime. This not only helps the students to understand the mechanism of RCP which will be the topic of the next lab, but also give the students a chance to understand efficient handling of code contribution with lazy class loading.

OSGi [7], an important basis of the Eclipse runtime, is not digged into. This is mainly due to the limited timeframe in our course. However, the improved documentation and development support for OSGi bundles since Eclipse 3.2 has drawn our attention and we are considering to include programming OSGi bundles in future courses.

Lab 8: Rich Client Platform This lab introduces programming rich client applications by defining extensions to the RCP [1].

Again, here we do not follow the usual procedure to put RCP application development after that of IDE extension. We admit that merely making extensions to the existing IDE, which is itself an RCP application, is easier to begin with than building an application on top of RCP. However, we feel it appropriate to follow our roadmap from ground up, as this reveals the underlying mechanism better. This is feasible because we have introduced the extension mechanism and the contribution of applictions in the previous labs.

RCP was initially introduced into Eclipse in v3.0. Version 3.1 and the latest 3.2 has made noticeable improvement for developing RCP applications by providing an improved new project wizard and RCP templates. We have seen these improvements greatly reducing the difficulty of learning RCP development.

Lab 9: Workspace and Resources This lab focuses on the Eclipse workspace and resource concept and APIs. Students are guided to develop a headless application that do statistics on the specified workspace.

Workspace is the central data model for the IDE. However, before introducing IDE extension, we would like the students to concentrate on the workspace and resource first. Fortunately, programs that work with the workspace do not even have to be part of the IDE, nor do they have to be a RCP or GUI program. Headless applications, for example the AntRunner application as part of Eclipse works on the Eclipse workspace and we are following this approach.

[1]To be specific, a rich client application build on top of RCP is called a RCP application in this paper.

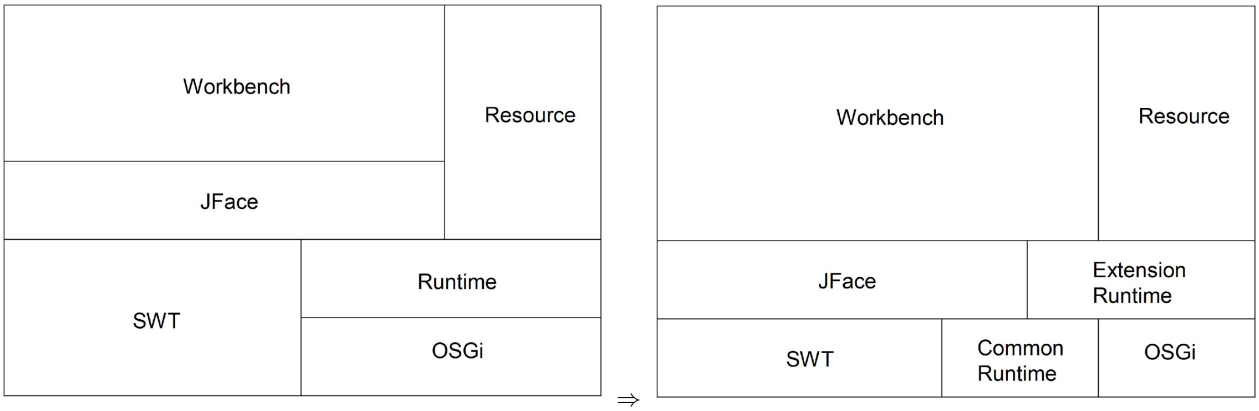

Figure 2: Split of Eclipse Runtime (Left: current fact; Right: imaginary split)

Lab 10: IDE This lab offer the students some hands-on practice in extending the Eclipse IDE. The lab starts with an explanation of the IDE as a RCP application, followed by an exercise in which the students continue their work in the last exercise to extend the Eclipse IDE for workspace statistics support.

As the last session in the hands-on lab exercises, this lab summarizes by giving the students some clue on where to find the necessary information when certain problems are encountered in their work. This includes the guidelines of plug-in programming, a summary of the Facade classes in the Eclipse runtime, platform, resource and IDE, as well as a summary of the references the students can consult for a specific type of problem. A list of the advanced topics that the students can discover themselves is also provided.

A single lab exercise is definitely not enough to cover even the most important aspects of IDE extension. However, on the one hand, with all the previous background of programming applications based on SWT, JFace, extending the Eclipse IDE is not much different from defining an extension to the RCP. On the other hand, the following mini projects give the students a real chance to become familiar with different aspects of Eclispe extension.

4. MISCELLANY

4.1 Background Preparation

Our experience shows that object-oriented programming in Java and certain aspects of software engineering like UML and design patterns are important background knowledge for our lab course. Students usually gradually grasp these knowledge in their first year studies.

For freshmen without previous knowledge in these aspects, we need to get them prepared as soon as possible. Therefore, depending on the background of the students, we give them an introduction to object-oriented programming with Java. This covers an overview of the main concepts of OO programming and of the Java language as well as of the compilation and runtime environment.

We also introduce UML as a vehicle for starting application development with a design phase. We believe that software design should be taught very early in a CS study program. While there is not enough time to cover all the ideas of software development with UML, we introduce only the main ideas and some of the main diagrams, i.e., the class diagram and the sequence diagram.

We use an UML plug-in (currently EclipseUML from Omondo [6]) for Eclipse to allow the students to familiarize themselves with UML and its integration into an IDE. The students learn that designing a software with UML is not wasting time but will automatically lead to appropriate source skeletons. Thus, the students can experience that working with UML enables easier interaction and information exchange with colleagues without requiring duplication of work.

4.2 Course Materials

All the course materials are available online [4]. Online materials include the presentations for the lectures, exercise sheets and links to the reference materials. The materials are available under the Eclipse Public License (EPL). A collection of screenshots of student projects are also available on the course web site.

In the spirit of open source, good tutorials from external sources that are publicly available are also used in the course. These include those from the Eclipse web site and the annual EclipseCon conference, as well as those from the IBM developer web site. We have also identified some good books, but they are references only for students in case they would like to go beyond the scope of our lab course.

In the development of this course, we find that although there are already papers, books, and online help available, the documentation for Eclipse is still not enough, especially for the need of learning and mastering Eclipse plug-in development. What is most needed, for example, include diagrams that describe plug-in dependency and extension relationships, and diagrams that describe each part of the Eclipse platform that can serve as rule of thumb when programming plug-ins. We expected to develop some of these documents during this course and will make them available once ready.

4.3 Teaching Techniques

E-learning techniques are utilized in the course. Technische Universität München, like many other Germany universities, has introduced Clix [1] as its central e-learning platform. One of the first courses of the computer science department that will be available on this platform will be our lab course. We plan to use Clix functionality for the following purposes:

- Registration of students.

- Distribution of teaching material based on a learning strategy. Course material for later exercises will only be available if the required prerequisites have been finished.
- Submission of solutions to exercises by the students.
- Giving feedback to the students on their solutions.
- Execution of tests with automatic correction to get an overview about the level of knowledge gained during the course.
- Establishing communication channels amongst the students and between students and teachers.

With Eclipse's internal Web browser, the students do not need to switch between Eclipse and an external browser to use Clix and work on the workspace. A complete integration of Eclipse and Clix demands both budget and experience, which will be gradually gathered in the process of our course.

Students participating in this course are requested to study a lot of background material. It is obvious that not all the background information can be presented by the teachers within 3 hours lectures. We assume that the students carefully read the referenced material and are able to extract the information required for the exercises. Therefore, the course is based on the students' ability to follow this learning technique. By targeting in this winter term the most talented high school students, we are optimistic that this requirement will be met by most of the students.

We do not use any Eclipse extensions that claim to make learning Java programming using Eclipse easier (for those interested, please refer to [8] for a summary of some of such extensions). Our course is so intensive that the students must learn Java and be familiar with Eclipse quickly and we have designed our course so that the learning curve is not too steep to follow. Of course, the students can appeal any tools that make their life easier. According to our experience, we haven't heard any students boaring about the difficulty of learning Eclipse usage.

5. SUMMARY

Teaching Eclipse plug-in development for undergraduate students is a demanding task, especially for first-year students.

Our initial attempt to teach Eclipse plug-in development for the undergraduate students, especially the first-year students, has proved the feasibility of offering such a demanding course to students of little or no previous knowledge of programming.

The course is especially suited for the most talented students since it offers insight into a number of very interesting areas of object-oriented programming. It covers basic oo programming techniques, UML modeling, pattern-based programming, familiarization with a huge code basis, designing and implementing graphical user interfaces with class libraries, understanding reusability concepts based on the extension mechanism, and implementing new interesting projects based on public domain software.

Based on our experience in the previous semesters, we are optimistic that the students will gain knowledge in the above mentioned areas that will enable them to understand more easily the benefits from the more theoretical base lectures taught in the undergraduate program.

6. ACKNOWLEDGMENTS

Thanks for IBM for supporting this course with Eclipse Innovation Grant 2005.

7. ABOUT THE AUTHORS

Tianchao Li is currently a full-time research associate at the computer science department of Technische Universität München, Germany. He is an Eclipse committer and has four years experience with Eclispe development. He has developed plug-ins for IBM, EP-Cache [5] project, Eclipse Parallel Tools Platform, and Globus Service Development Environment [3]. He is now supported by IBM Center for Advanced Studies to work on his Ph.D. studies on automated resource mangement for large-scale applications. Besides resource management and workflow support for the Grid and Eclipse-based tools for parallel and distributed computing, his research activities also include performance monitoring and tuning, computer architecture and its simulation.

Michael Gerndt is an associate professor for architecture of parallel and distributed systems at the computer science department of Technische Universität München, Germany. His research interests include language design, compilation techniques, and programming tools for parallel and distributed systems, automatic performance analysis for parallel programs. Michael is a member of the advisory board of the Euro-Par conference, the steering committee of the International Conference on Supercomputing (ICS), and the steering committee of the International Workshop on High-Level Programming Models and Supportive Environments (HIPS). He is also serving as a PC member of numerous conferences and workshops in the area of parallel computing. He has published more than 40 conference and journal articles in his area of research.

8. REFERENCES

[1] Clix e-learning platform. http://www.im-c.de/international/index.htm.

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

[3] Globus Service Development Environment. http://sourceforge.net/projects/gsde/.

[4] T. Li and M. Gerndt. Eclipse Plug-in Programming Lab Course Website. http://www.lrr.in.tum.de/~lit/teaching/eclipse.

[5] T. Li and M. Gerndt. Performance Cockpit: An Extensible GUI Platform for Parallel Tools. In *Proceedings of the 11th International Euro-Par Conference (Euro-Par 2005)*, Lishoa, Portugal, Aug. 30th - Sept. 2nd 2005.

[6] Omondo. EclipseUML. http://www.omondo.com.

[7] OSGi. http://www.osgi.org/.

[8] J. K. P. Bouillon. Using Eclipse in Distant Teaching of Software Engineering. In *Proceedings on Workshop on Eclipse Technology Exchange (eTX 2004)*, Vancouver, Canada, October 2004.