Adopting GILD:
An Integrated Learning and Development Environment for Programming

Margaret-Anne Storey[1], Mary Sanseverino, Daniel German, Daniela Damian,
Adrian Damian, Jeff Michaud, Adam Murray, Rob Lintern, James Chisan
Department of Computer Science
University of Victoria

Marin Litoiu
Center for Advanced Studies,
Toronto, IBM

Derek Rayside
Program Analysis Group
Laboratory of Computer Science, MIT

Abstract

This position paper presents GILD – an integrated learning and development environment for programming. The objective of the GILD project is to provide facilities for teaching and learning Java that are tightly integrated with a fully featured, mature and widely adopted development environment. GILD is being designed as a plug-in for Eclipse and takes full advantage of the Eclipse Java development tools. It will also include collaborative support as well as more sophisticated methods for teaching and learning the first principles of a programming language. In this paper we identify the technical and pedagogical aspects that we think will contribute to its adoption by both teachers and students, while discussing the challenges and barriers that we may face.

1. Introduction

Teaching students how to program can be a challenging task. Unfortunately, there are few tools that provide pedagogical support for learning and teaching programming. Fully featured integrated development environments (IDEs) overwhelm novice programmers and do not have all of features needed to support teaching. The programming pedagogical tools that do exist (such as BlueJ [15] and DrJava [14]) have not seen wide acceptance. We suspect this is because they tend to offer a minimal or reduced feature set and thus are limited in how long they remain useful to the student programmers. They also lack features that are commonly available in collaborative desktop environments, such as simultaneous document browsing/editing, instant messaging, web forums etc. Such features are often used in other disciplines to enhance learning.

Currently teachers usually make use of several tools when creating materials for a programming course. They may use an IDE to prepare program samples, a presentation tool to present materials in-class, a drawing tool to create pictures, a web-publishing tool to post course materials, and email to communicate with their students. This leads to a scattering of course materials and related information that is difficult to update and share. Moreover, the current approach of using email for providing programming help outside class time is very tedious and quickly breaks down – for example, students often ask questions about code that the instructor can't see. There exist some tools, such as Webboard and WebCT that provide a Web portal to the material of a course, but unfortunately these tools are not tailored to teaching and learning programming.

In our experiences, we have found that students will learn programming concepts more quickly if they read, write and test lots of example programs. Unfortunately instructors rarely have enough resources to provide feedback on numerous programming exercises. Automatic marking techniques are commonly deployed but they are typically custom solutions that are difficult to reuse or extend. Good teaching practices also recommend providing in-class interactive exercises – but with traditional pen and paper mediums it is impossible to give feedback to every student solution in large classes.

We believe there is a need for an integrated learning and development environment that will expand in usefulness as the programmer's ability and need for training increases. We have begun to develop such an environment to help novice through intermediate programmers learn Java. The GILD (**G**roupware enabled **I**ntegrated **L**earning and **D**evelopment) environment is being built on top of an existing and well-accepted IDE (integrated development environment) within Eclipse [3].

[1] Contact email: mstorey@uvic.ca; Website for this project: http://gild.cs.uvic.ca

Eclipse is an open-source platform for the creation of highly integrated tools [16].

Our environment will make use of the powerful infrastructure and large number of plug-ins that are provided by Eclipse and the Eclipse community. We are also using other tools and borrowing concepts from web-based learning tools and collaborative desktop technologies to enhance learning in both co-located and distributed settings. By building on top of a widely adopted and powerful integrated learning environment, while paying careful attention to pedagogical requirements, we believe the GILD environment could be widely adopted for teaching and learning programming.

The rest of the paper is organized as followed. In the next section we detail the adoption goals of our project, in particular emphasizing both technical aspects of the GILD environment as well as the pedagogical requirements that need to be met to ensure adoption. In Section 3, we discuss how we expect both students and instructors will adopt this tool and the expected benefits such a tool can provide. In Section 4 we discuss the adoption challenges that we face with both instructors and students, and from the educational institution's perspective. Finally Section 5 concludes this brief paper and summarizes our current and future work.

2. Research Goals

There are many issues concerning the adoption of a learning environment for programming. There are two sets of users for this environment – teachers and students. We conjecture that for both groups to adopt the environment, it needs to be inexpensive, easy to install, easy to use, and fit in with their existing tools and legacy course information. But more importantly, the tool needs to provide some gains with respect to the pedagogical needs of both groups. In this section, we first discuss some of the technical aspects of GILD and then explore the pedagogical objectives for both teachers and students and how they can be supported by the technical solutions we suggested.

2.1 Technical aspects

2.1.1 Implement GILD as a plug-in for Eclipse

The Eclipse environment and its Java development tools are already seeing widespread adoption. This rapid adoption was anticipated due to Eclipse's open source nature and its extensible architecture. Consequently we decided to build the GILD environment on top of the Eclipse Java Development Tools and Eclipse infrastructure. Moreover we will leverage third party plug-ins that can bring additional benefits to the users of the GILD environment. For example, there are plug-ins to support the automatic generation of UML diagrams (for example, the SlimeUML tool [1]). Generation of UML diagrams can help instructors explain code examples, or can be used by students during code explorations or to show their designs in course assignments. Other plug-ins of interest enable pair programming [2] and provide information on code quality [21].

The environment we build will also be available as an open source framework that is extensible (i.e. we will create GILD specific extensible points).

2.1.2 Integrate features from existing tools

From our own experiences as teachers and that of our colleagues, we are acutely aware that instructors struggle with integrating and synchronizing information from many different tools. Tool switching and synchronization problems are common to instructors of most fields, and hence the recent rise in popularity of Web-based learning tools [18]. Web-based learning tools provide an integrated environment for preparing and managing course materials. Although such an environment may seem to offer trivial improvements over using a selection of tools – the biggest advantage they offer is that of organizing course materials for both the instructors and students. Note that a key criterion for an effective teacher is to be organized [20].

Moreover, students should be able to use one learning environment for both software development and course material access.

Unfortunately, Web-based learning tools do not help as much as one would initially hope when teaching or taking a programming course. They are not tightly integrated with the development environments which are the cornerstone tool for both instructors and students in a programming course. To address this problem, we propose that key features from a Web-based course management environment should be tightly integrated with a software development environment (as opposed to the alternative approach of creating some programming support and adding that to a Web-based learning tool).

We propose that the User Assistance plug-in for Eclipse be used for authoring, integrating, storing, organizing and presenting Web-based materials. Such materials should be tightly coupled with the programming examples in the Java development tool repository. We are also exploring how collaborative support (currently being added to Eclipse for other projects) can be leveraged in the GILD environment.

The integrated approach we advocate would reduce the need for both students and teachers to be constantly switching between tools when interacting with a course.

2.1.3 Provide a customizable environment for learning and teaching

One size definitely does not fit all when it comes to teaching and learning programming. All instructors and students will have very different needs when using such an environment. Instructors have very different styles of teaching, and may choose to use only a subset of the features offered. The features they use in their course will also depend on the level of the course being taught.

We have noticed that even students in a first year programming course tend to be at very different skill levels and consequently the more advanced students will choose not to use the simple tools that are often advocated at universities as they are too limited. For example, at the University of Victoria, the TextPad tool [17] is recommended in first year as it is very simple and easy to learn for novice programmers. However, such a tool is clearly very limited and will be rejected by the more savvy students.

The instructor and the students have to be able to configure the environment so that it is suitable for the varied levels of the students. Eclipse offers many advanced features such as code assist and refactoring which may be overwhelming for novice programmers but desirable for the move advanced student. Customization of the features in the Eclipse JDT can be achieved in part through the use of its UI features (natures, perspectives and views), thus meeting the diverse needs of instructors and students while supporting changing needs over time for both groups.

2.2 Pedagogical objectives

2.2.1 Novice programmers should read, write and test lots of code

It is generally accepted that novice programmers should have lots of experience reading, writing and testing programs in order to learn. Unfortunately current teaching tools tend to place the students (and indeed the instructors) outside the domain of the development environment and instead trap them in a Web browser or in a presentation tool.

The GILD environment should instead position students and teachers within the development environment providing easy access to relevant executable program examples and other course materials.

We are using the existing version control systems in Eclipse (such as CVS) for storing examples and exercises selected by the instructor. Students will be able to check in/out code and assignments using these facilities. We will also integrate facilities to allow automatic deployment, submission and marking of assignments. Such facilities will enable students to do more programming (which is the best way to learn how to program).

Over time, a library of code examples, course notes and tutorials, can be created by leveraging and extending the features of the version control repositories and the help system in Eclipse.

2.2.2 Present syntactically and semantically correct code to students

Many novice programmers struggle when learning the basics of a new programming language. Their knowledge is very fragile, and seemingly innocent mistakes in an instructor's snippet of code, can cause students much grief. These mistakes are common when the program examples and code snippets are taken outside the development environment. By keeping such examples grounded within a development environment, the instructors can more easily correct syntactic mistakes as they occur in real time. Eclipse provides 'eager parsing' and 'code assist' features that can also be used to help the students learn from their own mistakes and may promote more exploration on the part of the students. We are also exploring these facilities to see if they lend themselves to customization. This would allow instructors to tailor messages to emphasize topics of relevance.

2.2.3 Assign Interactive Exercises in the classroom

Many universities have wired classrooms or laboratories where each student sits at an individual computer in a networked environment. In such an environment, a tool should provide support for the instructors and students to write, annotate and run code in real time–passing control from one to another as required. In addition, students should be able to submit answers that can be marked automatically and direct questions to the class for discussion as they arise.

We are also exploring how these objectives can be met through existing Eclipse plug-ins that support pair programming such as SanGam [2] and the collaborative support that is being added to Eclipse.

2.2.4 Provide support for Communication and Just-in-time Training

Web-based learning tools are in part also popular due to the collaborative support they provide. Communication mechanisms such as forums, instant messaging and email are used frequently. Such facilities take student interaction beyond the classroom and enhance the learning experiences of the students. Students can learn from and help one another when such facilities

are available. Without these, many students express isolation in a university environment and have only the instructor to approach when they have problems with course material. Moreover interactions with instructors can be limited to either office hours or to emails—which tend not to be very expressive.

We advocate that the students should be able to interact with the instructors and other students both in real-time and asynchronously by asking questions and receiving replies that are positioned *within the context of their code*. When the students and instructor are not co-located but are working synchronously, collaborative support features such as simultaneous code editing and instant messaging will move the interactions between the student and instructor out of the email world and back into the development world where these interactions should take place (see www.groove.net for an example of such a general purpose collaborative environment).

As the instructor receives questions about tricky parts of the assignment, the teacher can insert links to related code examples and other hints that will provide "just in time" training for the more complex exercises. Furthermore, pair-programming techniques have been used successfully in many introductory programming courses [19]. Simultaneous code editing combined with instant messaging will enable students and the instructor to collaboratively author code and improve their learning experiences.

Eclipse already has some infrastructure that we believe will be useful for helping us provide this support–such as extensible markers and decorators.

3. GILD's Adoption

There are many reasons that lead us to believe that GILD will be adopted. We list these reasons from four perspectives—that of instructors, students, post-secondary institutions, and the Eclipse community, while recognizing that these claims have yet to be validated.

3.1 Instructor Perspective:

Repository of course content. Using GILD we hope to achieve a closer integration of course materials (notes, pictures, animations etc) with executable program examples. Typically course content that one finds on the Web, or borrows from colleagues, is incomplete or lacks documentation. In particular, standalone programs are not often explicitly tied to course material or learning objectives and it takes a professor contemplating reuse of the material much time to figure out if the material matches their needs. By integrating such programs more closely with course material (linked by learning objectives) we hope to lead to more reusable content and programs. The reuse of lecture materials is very attractive to individual instructors and department administrations.

Fewer tools required when teaching. Such an environment could alleviate the need for switching and synchronization of materials between tools and hence lead to more adoption.

3.2 Student Perspective:

Popularity of Eclipse. Eclipse is already widely used in industry. Students will likely see the value of learning it, as they can apply their knowledge later when their education is finished.

Free. Students have very limited resources so cost will have a big impact on whether they adopt a tool or not. Gild will be free when used for academic purposes.

Collaborative support. Many students feel isolated from other students and indeed from the professor in a course. We conjecture that collaborative support would lead to more adoption of this approach.

Interactive learning support. Learning how to program is a very dynamic activity. As instructors of these courses, we have noted that the students that write lots of code and actively engage with the material do much better than students who take a more passive approach. Unfortunately, our current tools support the passive approach rather than a dynamic environment for code exploration and experiences. We believe students will embrace the opportunities to do more programming if they are presented to them in an easy to access manner.

3.3 Post-Secondary Institution Perspective:

Platform independence. GILD will run on any platform in which Eclipse runs. This includes Windows, Mac OS, and various flavours of Unix. Moreover, GILD will co-exist with other applications on these machines, as they are used by students, faculty and staff for many purposes.

Free. GILD, and all subsequent updates, will be free to post-secondary institutions. These institutions are under constant budget constraints, so even a small fee could be a break-point. As well, licensing issues can be easily dealt with. Students will also be able to use GILD free of charge on their own machines.

Easy to deploy and maintain. Typically, post-secondary institutions have limited system personnel and teaching assistance resources. At UVic, we currently use TextPad because it is easy to deploy and maintain. It is also easy to train teaching assistants and computer consultants on. However, TextPad is limited in that it is neither a learning nor a teaching environment. Our goal is to make GILD a tool that is easy to deploy and maintain,

easy to learn and use, and actively supports various learning/teaching strategies.

3.4 Eclipse Community Perspective:

Community oriented. We will build on top of other research and expect other research groups may wish to build on top of our work. We also expect several communities to flourish around GILD's repository. Instructors and students will be able to share their content and their experiences using GILD.

Open source. By making GILD open source we expect two main outcomes. First, it will be free for anybody to use. Second, other projects can reuse GILD or part of it in order to provide functionality or products beyond the original goals of the project.

Extensible. Our objective is to create an architecture that permits the customization of GILD to specific environments, for example, teaching C/C++.

4. Adoption Challenges

As in many areas of technologically influenced change, the adoption challenges for GILD are classic. They include the following: infrastructural capability, staffing/training issues, and, perhaps most importantly, attitudinal differences in the potential adopter community.

Infrastructural capability: In the post-secondary education community, change in technological infrastructure for teaching is often determined by committee. Good teaching/learning rationales have to be provided to these committees before any technology that "pushes the infrastructure envelope" is adopted. Typically, those responsible for day-to-day teaching infrastructure are the first to recognize the benefits of such upgrades, and are often looking for "good causes" to support their requests. Of course, any infrastructural change must be manageable for the target institution(s). Moreover, most post-secondary institutions have commitments, both formal and informal, to supporting widely varying technologies within the same infrastructure. Technologies that need to "take over" existing infrastructure are often not successful in this milieu. GILD should balance as small an infrastructural change footprint as possible while providing exceptional teaching/learning possibilities.

Staffing/training issues: The typical post-secondary computer help desk is a hive of activity – especially when assignments are due. Staff are usually run off their feet. Therefore, to be readily adopted, new technology must be robust enough to run without much intervention by skilled staff. However, in GILD's case, we do want users to ask questions, but about content, not operation. Therefore, front line staff have to be trained to use GILD and be kept up to date on the types of questions to expect from users. Moreover, GILD will need to be easy to install and run on users' home machines. We could anticipate that some demands may be reduced on teaching assistance staff, as it will be easier for students to help one another 24/7. As instructors, we have noted a big decrease in student questions when we provide facilities to support their interactions outside the classroom.

Attitudinal differences: In most post-secondary institutions, the resources used to teach a given course are very much influenced by the preferences of the individual instructor. Any new learning/teaching technology would have to easily interface with the majority of the resources already used by the instructor. Failure to do so would be a major barrier to adoption. Certainly many instructors want to improve their teaching methods, but almost every instructor has already spent a great deal of time preparing and testing material. The GILD system must be able to integrate this legacy material and provide instructors with new ways of using and building on it.

Other challenges: We have yet to discover what other challenges and barriers to adoption there remain with respect the Gild tool. We look forward to feedback at the Adoption Centric Software Engineering workshop on people's opinions and insights about our proposed work. For a tool to be adopted, it must fill a need and provide advantages that outweigh the disadvantages from adopting such a tool. Do the needs we identified resonate with others in software engineering? Does it seem likely that we can overcome the challenges and reduce the potential adoption barriers we identified? And are there other significant challenges that we may face that we have not yet considered?

5. Conclusions

GILD is an integrated learning and development environment for programming. Our goals for this environment are to improve the experiences of students' learning and professors' teaching Java programming. Besides the more general adoption challenges, GILD faces challenges of providing gains with respect to the pedagogical needs of both students and teachers. In this paper we described a project in which we intend to overcome these adoption challenges by making use of the powerful infrastructure and large number of plug-ins available in Eclipse, as well as by thoroughly researching the pedagogical needs that such an environment provides.

Our project is in its early stages, where our challenges include the rigorous definition of the technological as well as pedagogical needs of the intended GILD users. We are currently striving to provide technical solutions to these needs as well as creating a research environment in which the adoption barriers are well-understood and addressed through proven research methods. The user requirements

will be defined through iterative prototyping with intended categories of students and teachers, while the environment will be user tested following its development. We also intend to address adoption by organizing workshops and training with GILD and observe its use in classrooms for continued improvement and removal of adoption barriers.

Developed through well-identified research methods, we expect the GILD project to provide a powerful tool to convey our research practices to other disciplines and to advance research in our community. It will foster continued collaborations within the Eclipse community as well as among researchers in the area of adoption-centric software engineering.

References

[1] Slime UML plugin, http://www.mvmsoft.de/content/plugins/slime/slime.htm

[2] Pair Programming Plug-in, http://sourceforge.net/projects/sangam

[3]Eclipse Overview, http://www.eclipse.org/whitepapers/eclipse-overview.pdf

[4] Eclipse Documentation (pdf format):

[5] Eclipse perspectives, http://dev.eclipse.org:8080/help/content/help:/org.eclipse.platform.doc.user/concepts/concepts-4.htm

[6] Eclipse Views, http://dev.eclipse.org:8080/help/content/help:/org.eclipse.platform.doc.user/concepts/concepts-5.htm

[7] Natures, http://dev.eclipse.org:8080/help/content/help:/org.eclipse.platform.doc.isv/guide/resAdv_natures.htm

[8] Refactoring, http://dev.eclipse.org:8080/help/content/help:/org.eclipse.jdt.doc.user/reference/ref-115.htm

[9] CVS, http://dev.eclipse.org:8080/help/content/help:/org.eclipse.platform.doc.user/reference/ref-47.htm

[10] Code assist, http://dev.eclipse.org:8080/help/content/help:/org.eclipse.jdt.doc.user/reference/ref-143.htm

[11] Eclipse Markers, http://dev.eclipse.org:8080/help/content/help:/org.eclipse.platform.doc.isv/guide/resAdv_markers.htm

[12] Team Decorators, http://dev.eclipse.org:8080/help/content/help:/org.eclipse.platform.doc.isv/guide/team_ui_decorators.htm

[13] Help System, http://dev.eclipse.org:8080/help/content/help:/org.eclipse.platform.doc.user/tasks/tasks-1g.htm

[14] DrJava: A lightweight pedagogic environment for Java, Eric Allen, Robert Cartwright, and Brian Stoler September 7, 2001 ,Presented at SIGCSE 2002

[15] David J. Barnes & Michael Kölling, Objects First with Java A Practical Introduction using BlueJ Prentice Hall / Pearson Education, 2003, ISBN 0-13-044929-6

[16] Eclipse FAQ, http://www.eclipse.org/eclipse/faq/eclipse-faq.html

[17] Texpad, http://www.textpad.com/

[18] Evaluating the usability of Web-based learning tools, M.-A. Storey, B. Phillips, M. Maczewski and M. Wang, Special issue on Evaluation of Learning Technologies in Higher Education, Guest Editor: Grainne Conole, *Educational Technology & Society 5 (3) 2002, ISSN 1436-4522*

[19] In support of student pair-programming , Laurie Williams, Richard L. Upchurch , Technical Symposium on Computer Science Education, Proceedings of the thirty second SIGCSE technical symposium on Computer Science Education 2001 , Charlotte, North Carolina, United States

[20] Effective Teaching Behaviors in the College Classroom. Harry G. Murray in Effective Teaching in Higher Education: Research and Practice. Editors: Raymond Perry, John Smart. Agathon Press, New York. 1997, pgs. 171-204

[21] Eclipse Metrics Plug-in, http://www.teaminabox.co.uk/downloads/metrics/