GILD:
An environment to facilitate collaboration in teaching and learning

Dr. Margaret-Anne Storey
The Chisel Group,
University of Victoria,
British Columbia, Canada
University of Victoria: 
  Margaret-Anne Storey, Daniela Damian 
  Mary Sanseverino, Daniel German, Del Myers, 
  Elizabeth Hargreaves, Glen Kiessling, Jeff Michaud, 
  Adrian Damian, Robert Lintern 
IBM:  Marcellus Mindel, Marin Litoiu 
Dalhousie University:  Karen Parker 
MIT:  Derek Rayside
Motivation for the project

• Many students *struggle* in the first year and experience feelings of *isolation*
  – Good methodologies for teaching programming involve significant *hands-on* experiences
  – Some *pedagogical tools* do exist (DrJava, BlueJ) but they are limited
  – *Pair Programming* can be very powerful for students [Williams]

• *Lack of* tools to provide support for both the instructor and the learner
  – An instructional tool that will *grow* with the student but is *powerful* for the instructor
  – Need to support *active learning*
  – Content management (enable *reuse, sharing*)
Objective

_to improve teaching and learning in introductory programming courses through improved technology_

Proposed solution: Develop a plug-in for Eclipse that will provide cognitive support for novice programmers and support for instructor’s activities
Project Goals

• **Research Goal:**
  – Investigate if we can improve learning and teaching of programming by providing a flexible learning and development environment that facilitates collaboration and community building

• **Engineering Goal:**
  – Build an integrated system to provide support for both students and teachers using Eclipse and its plug-ins
General Methodology

• Iterative phases of:
  – Requirements gathering from students, instructors, teaching assistants, administrators at UVIC and other institutions
  – Design and implementation
  – Deployment and evaluation -- locally at first and then beyond
Current Work

• Extensive requirements gathering through:
  – Student surveys
  – Faculty and Teaching Assistant focus groups
  – Interviews with lab administrators

• Development of Gild 1.0:
  – Student perspective
  – Instructor and teaching assistant views and tools

• Evaluation:
  – User studies of preliminary prototypes
  – Heuristic evaluation
  – Deployment (Spring 2004):
    • Second level programming course, approx. 360 students in three sections
    • Questionnaires
    • Student observations (with permission) and focus groups
Requirements for Students

- Integrated learning and development environment for *students*:
  - *Simpler* development tool (initially) that provides access to code *examples* linked with other course materials such as lecture notes
  - *Pedagogical support* for programming concepts
  - Support for *collaboration + communication* with students and teachers (in and outside the classroom, synchronously and asynchronously)
Requirements gathered for the Simplified Student Perspective

- Which features to *keep*
- Which features to *remove*
- Which features to *simplify* and *enhance*
- Which new *features* can we add to provide additional cognitive support for student learning
### Simplified student perspective -- Features

<table>
<thead>
<tr>
<th>Category</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Core Features to include</strong></td>
<td>Compile and launch operations; Syntax highlighting; Line numbers (on by default); Debugger; Task view; Resource view; Project import/export</td>
</tr>
<tr>
<td><strong>Features to exclude</strong></td>
<td>Wizards (some of); Refactoring</td>
</tr>
<tr>
<td><strong>Features that are simplified</strong></td>
<td>One-click launch; Debugger with single thread and step over features; Project import/export simplified; Task view shows only errors for current project by default</td>
</tr>
<tr>
<td><strong>Features that are enhanced</strong></td>
<td>Resource View: decorators on resources to indicate file state and distinction between .class and .java files; .class files</td>
</tr>
<tr>
<td><strong>New features to support learning needs</strong></td>
<td>HTML browser with features to link to and dynamically include source code; integration of Dr Java Interactions pane</td>
</tr>
<tr>
<td><strong>Support different teaching and learning styles</strong></td>
<td>Configurable settings for Gild: Use of debugger, debugging filters, CVS, automatic build, Java creation wizards; tooltips, line numbers etc.</td>
</tr>
</tbody>
</table>
Requirements for Instructors

- More integration of tools for instructors:
  - Content management -- support for building and maintaining code examples linked to conceptual learning materials
  - Collaboration with other faculty and Communication with students
  - Deployment of course deliverables and Assignment marking (marking perspective)
  - Support for interactive exercises in the classroom
  - Customizable (to support different teaching philosophies)
Implementation of Gild 1.0

- **Student perspective**
  - Simplified menus, run, resource view
  - Simplified debugger
  - Additional cognitive support (e.g. leverage decorators)
  - Preferences
  - Gild web browser

- **Instructor perspective**
  - Gild web browser
  - Marking
Gild – simplified user interface

```java
public class ColourClass {
    String colour;
    public void printColour() {
        System.out.println("The colour of this class is: " + colour);
    }
    public static void main(String[] args) {
        ColourClass colour1 = new ColourClass();
        colour1.colour = "red";
        ColourClass colour2 = new ColourClass();
        colour2.colour = "yellow";
        System.out.println("The colour of this class is: " + colour1.colour);
        System.out.println("The colour of this class is: " + colour2.colour);
        System.out.println("Hello World and Note we have one class, but two instances (objects) of that class!");
        colour1.printColour();
    }
}
```
Gild decorators in the Resource view

Loops.java needs to be saved

Loops.java needs to be built
**.class files**

1/  
2  ************ You have selected a .class file ************
3  .class files are binary files that contain computer-
4  executable code. They cannot be edited, but here is
5  an overview of the source code represented by the file.
6  
7  Class name:
8  PowerOfTwo

9  Fields:
10  
11  powerOfTwo

12  Methods:
13  
14  <clinit>
15  <init>
16  main
17  getNextPower

20 \----------------------------------------------------------------------/
Simplified Debugger

```java
public class Loops {

    public static void main(String[] args) {
        runWhile();
        runFor();
    }

    /**
     * Counts to 10 using a while loop.
     */
    public static void runWhile() {
        System.out.println("Counting up using while");
        System.out.println(" ");
        int i = 1;
        while (i <= 10) {
            System.out.print(i +", ");
            i = i + 1;
        }
        //Question: what is i at this point?
        System.out.println("done.");
    }

    public static void runFor() {
    }
}
```
Customizable features

• Two reasons for Gild specific preferences:
  – To suit different pedagogical styles
  – To provide a transition path for students to the fully featured Eclipse JDT

• Preferences:
  – Use of debugger
    • Debugging filters
  – CVS
  – Automatic build
  – Tooltips, line numbers
  – Java creation features
  – …..
**New features**

- *Gild Web browser* with special HTML tags that:
  - Provide links to the code in Eclipse
  - Dynamically include code from source code files
- Integration of *Dr. Java* interactions pane
  [www.drjava.org]
- *Marking annotations* provided by teaching assistants (prototype)
Prototype of marking tool

public class Loops {
    public static void main(String[] args) {
        runWhile();
        runFor();
    }
    /**
     * Counts to 10 using a while loop.
     */
    public static void runWhile() {
        System.out.println("Counting up using");
        System.out.println(" ");
        int i = 1;
        while (i < 10) {
            System.out.print(i + ", ");
            i = i + 1;
        }
        //Question: what is i at this point?
    }
}

EclipseCon 2004: Eclipse At work
Implementation experiences

- Support in Eclipse
  - Easy to add features through extension points
  - Good documentation and source code
  - Good resources (newsgroups, etc.)
  - Leveraging other plug-ins from the Eclipse community

- Hurdles (worked with version 2.1 only)
  - Difficulties removing features
  - Difficulties setting defaults
  - Difficulties accessing some useful APIs
Deployment experiences and early feedback

• Students
  – Installation challenges
  – Attitude generally favorable towards Gild and Eclipse
  – Recognize the importance of Gild learning units
  – More opportunities for programming (through provided examples)
  – Response to the interactions pane favorable
  – Some features in Gild and Eclipse still confusing
  – Quotes….
Early Quotes....

“I wrote all this code, and Gild keeps giving me all these errors”

“I want to use Content Assist but I want to use the Gild perspective”
Deployment experiences and early feedback

• Instructors:
  – Very powerful for content management and deployment of content (but some tasks are very difficult to do -- too easy to make mistakes)
  – Collaborations with students and instructors supported through CVS and common structures of the Gild projects (lacks support for content structures and relationships)
  – Interactions pane and scrapbook help during class presentations of course concepts
  – Benefits of task view (both for personal and assigned student tasks)
  – Eclipse Hierarchy view -- complicated for students, but instructors can leverage it during teaching
Learning Objectives for this week

- Review of Csc 110 material:
  - Know the difference between Classes and Objects
  - To be able to define a class with variables and methods
  - To be able to instantiate objects
  - To understand how to create simple constructors
  - To be familiar with the main method and how to use it for testing each class you create
  - Be familiar with primitive types and how to allocate storage for them
  - To gain preliminary familiarity with the String class and how to use it
  - To learn naming conventions in Java

- Introduction to Gild and Eclipse (as part of the first lab)

Learning Resources for this unit

- Sample programs as follows in the examples directory of the Gild project:
  - ColourClass.java
  - Point.java
Deployment experiences and early feedback

• Instructors:
  – Very powerful for content management and deployment of content (but some tasks are very difficult to do -- too easy to make mistakes)
  – Collaborations with students and instructors supported through CVS and common structures of the Gild projects (lacks support for content structures and relationships)
  – Interactions pane and scrapbook help during class presentation (demonstrations of course concepts on the fly)
  – Benefits of task view (both for personal and assigned student tasks)
Eclipse Task View and Gild
Deployment experiences and early feedback

• Instructors:
  – Very powerful for content management and deployment of content (but some tasks are very difficult to do -- too easy to make mistakes)
  – Collaborations with students and instructors supported through CVS and common structures of the Gild projects (lacks support for content structures and relationships)
  – Interactions pane and scrapbook help during class presentation (demonstrations of course concepts on the fly)
  – Benefits of task view (both for personal and assigned student tasks)
  – Eclipse Hierarchy view -- complicated for students, but instructors can leverage it during teaching
The Eclipse Hierarchy View
Deployment experiences and early feedback

• **Administrators:**
  – Installation of Eclipse is relatively easy
  – Cross platform a big win
  – Issues with storing and saving student workspaces
  – System requirements a bit steep
Current Challenges

• Content management -- looking at the Eclipse Training Perspective
• Visualization and animation support – the Penumbra tool has a simplified hierarchy view
• How to transition from Gild Java editor to the default Eclipse Java editor
  – How to support a student’s “optimal experience of being in the flow” as they learn more
• Being “in the flow”
  – the ability to concentrate on a task
  – immersed and engaged to the point where the perception of time slows
• Staying “in the flow”
  – the task must be neither too difficult to be discouraging, nor so easy that it is boring
  – unnecessary interruptions will destroy the feeling of flow
  – flow is improved when users spend more time on the task domain, and less time on the interface domain
Opportunities and Future Work

- Gathering requirements for next version during deployment (focus on collaborative aspects)
- Design new features based on requirements
- Leverage other plug-ins for Eclipse and other pedagogical tools for teaching programming
  - e.g. BlueJ
  - Integrate animation tools (for familiar algorithms, data structures)
  - Leverage plug-ins that do more analysis (e.g. the PMD plug-in for code quality evaluation)
  - Testing -- JUnit
Collaboration opportunities

- Collaboration between students
  - Pair programming (e.g. leverage the Sangam plugin)
  - Chat, email, team tasks
  - Anonymous peer review (critical thinking)
- Collaboration between instructors
  - Improved content management and navigation support through shared tags, visibility of comments, solutions etc
  - Community-based set of resources (e.g. http://eclipse.org/ecesis)
- Collaboration between instructors/TAs and students
  - Deploying the marking feature later this term -- needs to be improved, we will gather requirements for future versions

- Interested to collaborate with other researchers in this area!
Discussion....

For more information:
– Email: mstorey@uvic.ca

Ecesis project: http://eclipse.org/ecesis

Supported by:
– Eclipse Innovation Grants
– NSERC
– University of Victoria
References (1)


• “Strengthening the Case for Pair Programming”, Laurie Williams, Robert R. Kessler Ward Cunningham, Ron Jeffries, IEEE Software.


References (2)

- DrJava project, Rice University, [http://drjava.org/](http://drjava.org/).
- Ben Bederson: “Interfaces for Staying in the Flow”