GILD

Groupware enabled Integrated Learning and Development for Java

http://gild.cs.uvic.ca/

Discussion

- Is teaching in CS different to other disciplines, if so how?
- What techniques have you been exposed to when learning how to program in first year?
- Do you have suggestions for how technology could have been used to improve your learning?

GILD Project Overview

- Eclipse plug-in as instructional technology
- Improve learning for students
  - Simpler development tool (initially) that provides access to code examples linked with other course materials such as lecture notes
  - Support for collaboration + communication with students and teachers (in and outside the classroom, synchronously and asynchronously)
- Improved tools for instructors
  - Support for building and maintaining code examples linked to other course materials
  - Communication with students
  - Deployment and submission of course deliverables

Motivation for the project

- Java is the first programming language taught at many universities but few tools exist to help instructors teach and students learn how to program
  - Some exceptions: DrJava and BlueJ
- Good methodologies for teaching students how to program involve significant “hands-on” experiences
- “Pair Programming” has been shown to be a very successful technique in the classroom
- Instructors currently get bogged down using many different tools
- Need an instructional tool that will grow with the student but is powerful for the instructor

Dr Java *(built using Dr Java)*

drjava
A Lightweight IDE

Two Panes (third for file mgt):

- Interactions Pane
  - Immediately see results of expressions & statement
  - Alternate entry points (testing simpler, no main(), no debugger to learn)
  - Interactively create GUI
- Definitions Pane (code)
  - Brace matching, syntax highlighting, automatic indenting, pretty printing
  - Compiler integration

Diagnosis: Needs work, better support, more tuning

BlueJ

- Premise: Environments for OO are not OO environments; they do not reflect the paradigm
- BlueJ is object-oriented
- Designed for teaching
- Downsides:
  - Student must learn both java and protocols for BlueJ
  - Does not scale to large systems (at all)
  - Students must eventually move to Java text after all

BlueJ supports

- graphical class structure display
- graphical and textual editing
- built-in editor, compiler, virtual machine, debugger, etc.
- easy-to-use interface
- ideal for beginners (but unsuitable for pros)
- interactive object creation
- interactive object calls
- interactive application development and testing

Why are we interested in this project?

- Experiences teaching first year courses
  - Lack of tools to provide support for both the instructor and the learner
  - Many students struggle in the first year
  - Dwindling resources to support instructors at most universities
- Many interesting research issues
- Possibly many applications beyond 1st year programming
- Eclipse Innovation Grant
  - Dec 2002 (just started working on this project in Jan)

Status so far

- Background research
  - learning theories; psychology of programming
  - Eclipse plug-ins
  - collaborative technologies
  - pair programming
  - research methodologies (qualitative, quantitative)
- workshop (in mid February)
  - brainstorming
  - thought about goals for next six months

Dimensions of the project

- Collaborative support
  - CSCW, CSCL
- Organization, Management
- Education
  - Psychology, Sociology, Education/Curriculum
- Software engineering
  - HCI
- Content Management
  - Hypermedia

Broad Goals and Objectives

- Improve life for Students
- Improve life for Teachers
- Build a high-quality innovative tool
- Build a tool and research program to facilitate the exploration of interesting research questions while collaborating with other groups in the Eclipse community

Broad Goals and Objectives– more detail

- Improve life for Students
  - A tool to support development and learning (grows with the student)
  - Student-Student, Student-Teacher collaborative support
  - Adaptive help
- Improve life for Teachers
  - Course Management
  - Integration of lecture notes with code examples
  - Reuse of course materials (within and across universities)
  - Provide a unified approach to assist TA's and instructors

Broad Goals and Objectives cont.

- Build a high-quality innovative tool that is
  - Extensible
  - Portable
  - Deployable (staying within system constraints)
  - Used outside of UVic (widely adopted – top-10 eclipse plugin/google result)
- Work on interesting research questions
  - Tool should facilitate research
  - Papers
  - Theses
  - Collaborations
  - Learn new technologies

Refined Goals (6-9 month period)

- improve learning of Java in intro courses
- improve teaching of Java in intro courses
- integration of lecture notes and code
- homework deployment and collection
- support for pair programming

What is pair programming?

TWO programmers working side-by-side, collaborating on the same design, algorithm, code or test. One programmer, the driver, has control of the keyboard/mouse and actively implements the program. The other programmer, the observer, continuously observes the work of the driver to identify tactical (syntactic, spelling, etc.) defects and also thinks strategically about the direction of the work. On demand, the two programmers can brainstorm any challenging problem. Because the two programmers periodically switch roles, they work together as equals to develop software.

-- **Laurie Williams**
North Carolina State University Computer Science

KNOWLEDGE IS commonly socially constructed, through collaborative efforts towards shared objectives or by dialogues and challenges brought about by different persons' perspectives.

G. Salomon (book: *Distributed Cognitions: Psychological and Educational Considerations)*

What is pair programming cont.

- Think of a good pair driving across the country. One will drive, the other navigate (thinking tactically and strategically)
- Often used as a part of extreme programming

My own experiences teaching 1st year

- Bimodal distribution of scores on the midterms
- Some students seem very “cocky” – others are convinced they are “terrible at programming”
- Very little resources to help students
- Self-esteem quite low for many
- Isolation is prevalent among many of the students

My own experience with pair programming

- Setting the stage is very important (did exercises in class)
- Students for the most part loved it and said they would do it again but a few hated it…
- Matching of skills is probably important…. They seemed to think so
- Biggest problem experienced by students was finding a common time to get together at school (some work etc)

Code Warriors and Code-a-Phobes

- Code warriors see themselves "*as a sort of code-warrior,fighting with the enemy compiler, forcing it to assent to their glorious code and to produce a program that obeys their every desire"*
- Code-a-phobes – seems to be an unfortunate phenomenon in computer science, report that they "*hate programming"* or that they are "*hopeless at programming*"
- Mixture of such students is part of the challenge of teaching first year programming

Study on use of pair programming

- Williams' studies indicate about 80-90% of students like pair programming and feel their solutions are more correct
- But another study showed differences between code warriors and code-a-phobes
  - Two variables – attitude and performance, may be independent
  - Better if they are matched in similar pairs w.r.t. attitude (didn't look at performance in this study)
  - Code warriors are less likely to enjoy pair programming

Discussion points

- Can we create some kind of virtual environment to enable pair programming at distributed locations – or would that not remove the condition that makes it so special?
- How can we make first year programming more fun and interesting? Can technology help?
- How can we build self esteem?

Leveraging existing components in Eclipse

- A look at how we can leverage and extend the existing technologies in Eclipse:
  - Perspectives
  - Eclipse help system and other User Assistance (UA) technologies
  - Content Assistance in Eclipse
  - Markers in Eclipse
  - Existing visualizations plug-ins (UML, SHriMP, Pair programming)
- Furthermore, many more plug-ins that are being added to Eclipse could prove beneficial
  - Other groups working on related projects (e.g. Echelon, Cool project in France)

Perspectives

- What is an appropriate perspective for novice programmers?
- User studies would need to be done to discover the ideal set of views and features for novice programmers – as well as consulting instructors

Eclipse Help System

- The help system has (or will have) support for being served from a web server
- The appearance of the help browser can be customized
- Integration of documentation sources is supported
- Suport for other content types such as DHTML, Flash, Javascript, XML
- Search Engine functionality can be extended
- Infopops and Active help

Content Assistance

- Could be tailored to novice programmers
- The amount and type of content assist would vary as the programmers learn the concepts
- The tool would need to "remember" using very simple knowledge management techniques what the programmer has learned
- Eventually the content assistance provided would be the same as provided to expert users

Markers in Eclipse

- Existing:
  - **Problems** - for representing invalid states (errors, warnings, information)
  - **Tasks** - for capturing user created reminders (todo's)
  - **Bookmarks** - for marking a location that can be quickly jumped to later
- Could be extended with new marker types to support learning tasks – use of additional metadata to support sharing and searching

Eclipse Decorators

- Visual cues, useful state information
- 2 types
  - Text label decorations (prefix and suffix)
  - Image decorations (superimposed image on an icon)
- GILD Decorators?

Eclipse Project Nature

- Concrete link between a project and tools or feature set
- Ex. Java Nature > JDT
- Determines lifecycle of tool's interaction with a project
- Icons, actions, etc, reflect project nature
- Can control under what circumstances a project nature enabled

Koi

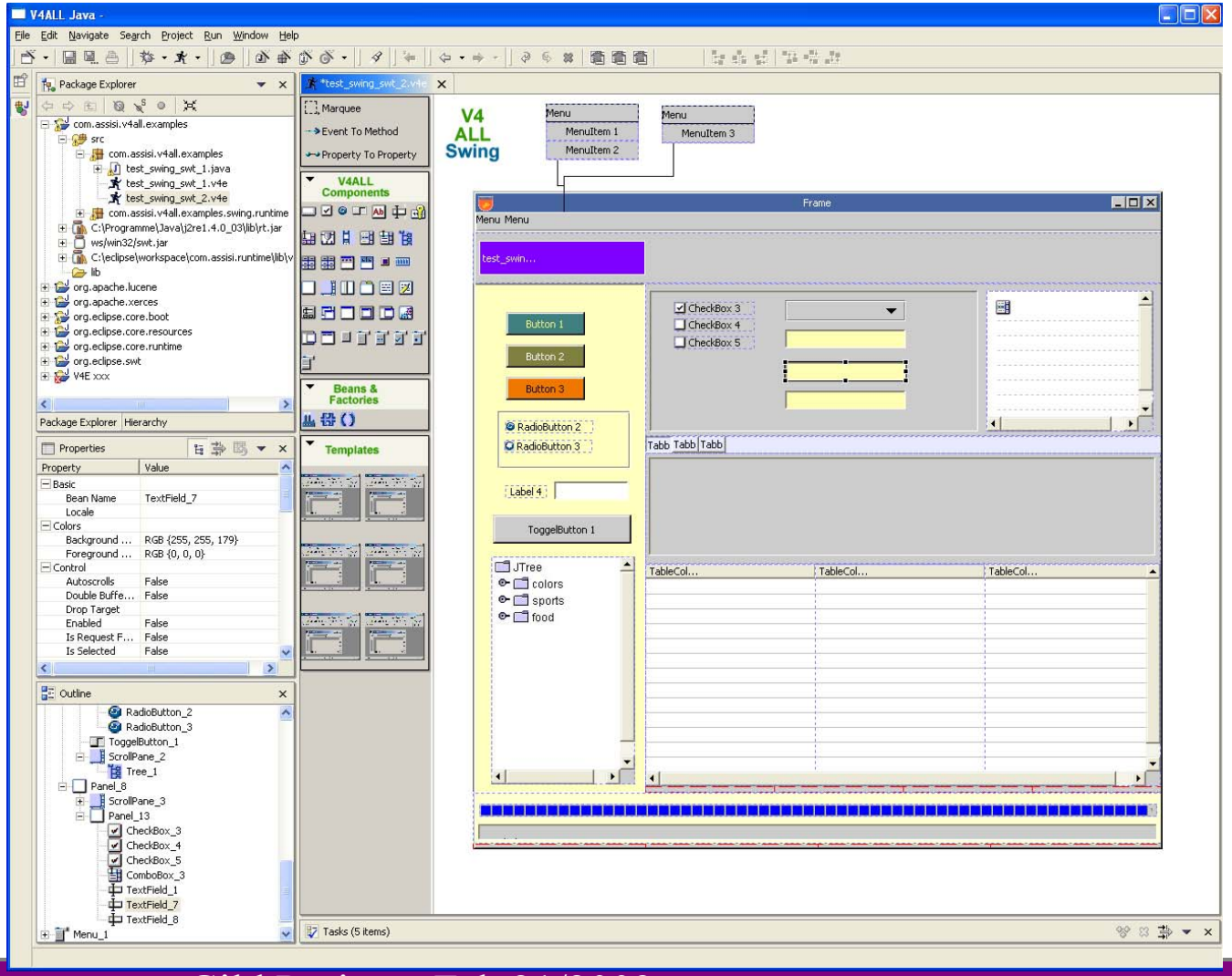
- Collaborative Infrastructure for Eclipse (?)
  - Connections of pieces through *extensions*
- client/server (JDBC DB for persistence)
- Web Services RPC communications model > SOAP
- core services: user, permission, message, storage
- Good docs, but “prototype”
- Server security = “none”

Plugins

- There are currently over 230 Plugins
- Look at several Plugins that have potential for use within GILD
  - Inspiration
  - Direction
  - Learn from their mistakes

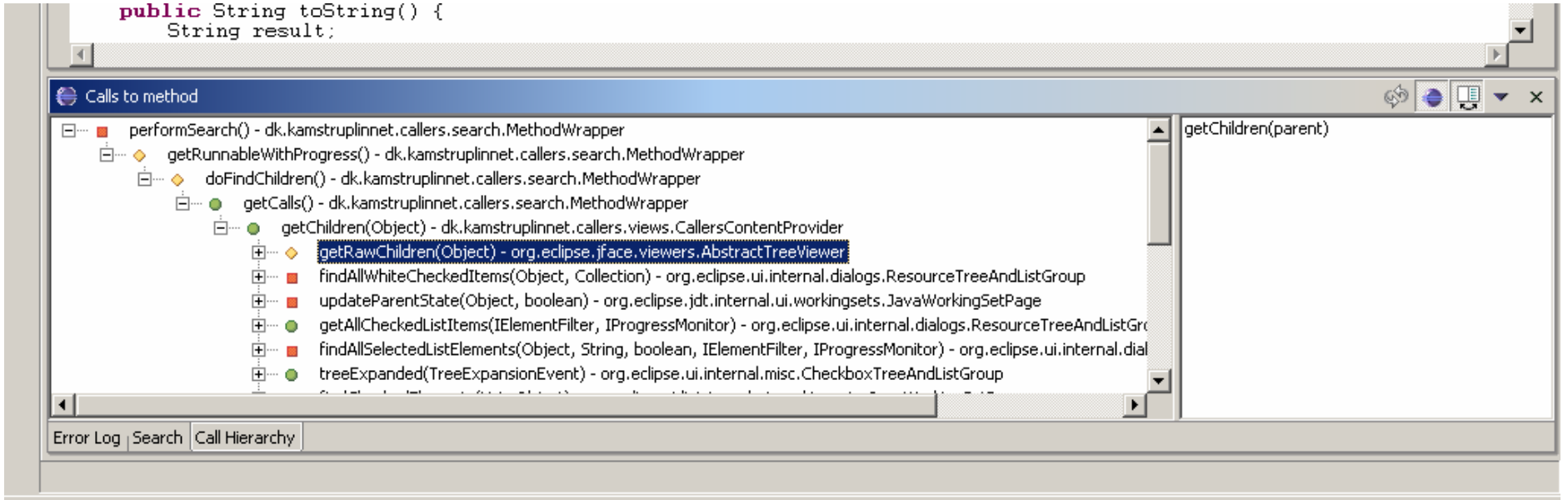
Assis Eclipse GUI

- GUI Builder (with SWT)
- 2 months on Eclipse Plugin List
- Very Active



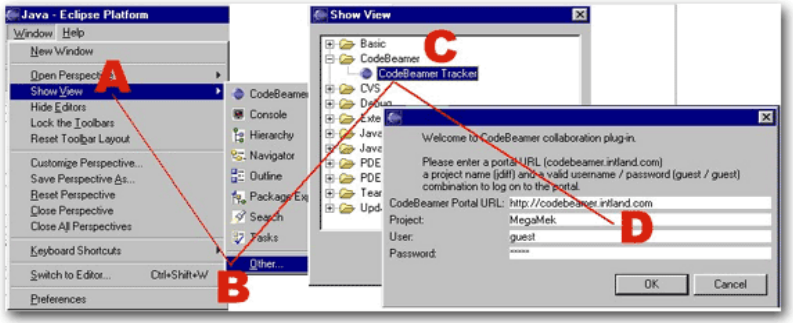
Call Hierarchy View

- This plugin features a Call Hierarchy view which can show calls to or from a method in a tree
- About 1 month on plugin list



CodeBeamer

- Group Management with web interface
- Commercial
- Database/CVS backend
- Look to inspiration
- About 10 months on plugin list

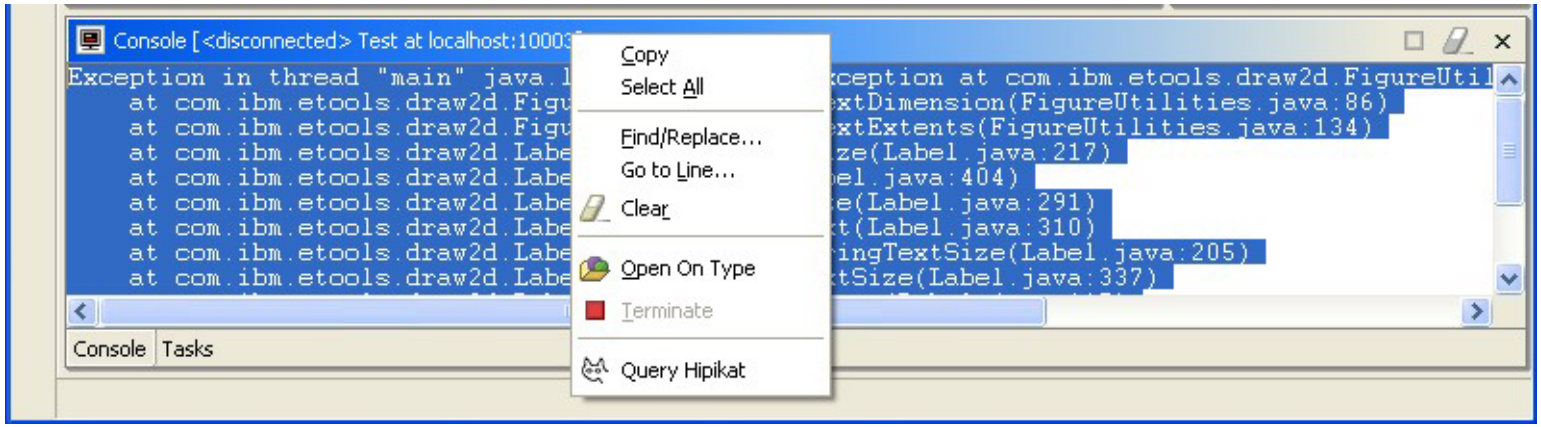


EclipseMetrics

- Out of range metrics cause additions to task list
- Supported Metrics
  - McCabe's Cyclomatic Complexity
  - Lack of Cohesion in Methods
  - Number Of Fields
  - Number Of Levels
  - Number Of Parameters
  - Number Of Statements
  - Weighted Methods Per Class

Hipikat

- Hipikat ***recommends*** relevant software development artifacts based on the context in which a developer requests help from Hipikat
- Repository consists of entries in CVS, Bugzilla, Newsgroups/Emails
- 6 months on Eclipse list

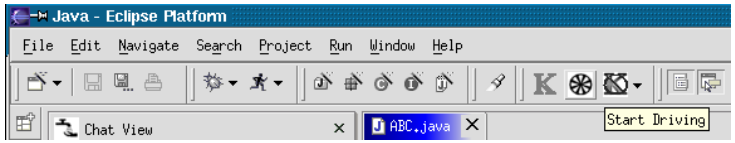


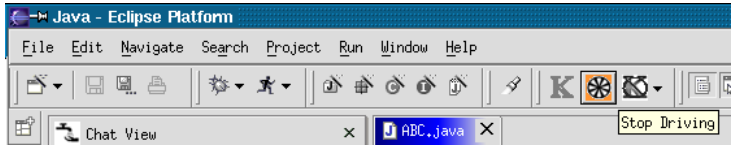
PMD

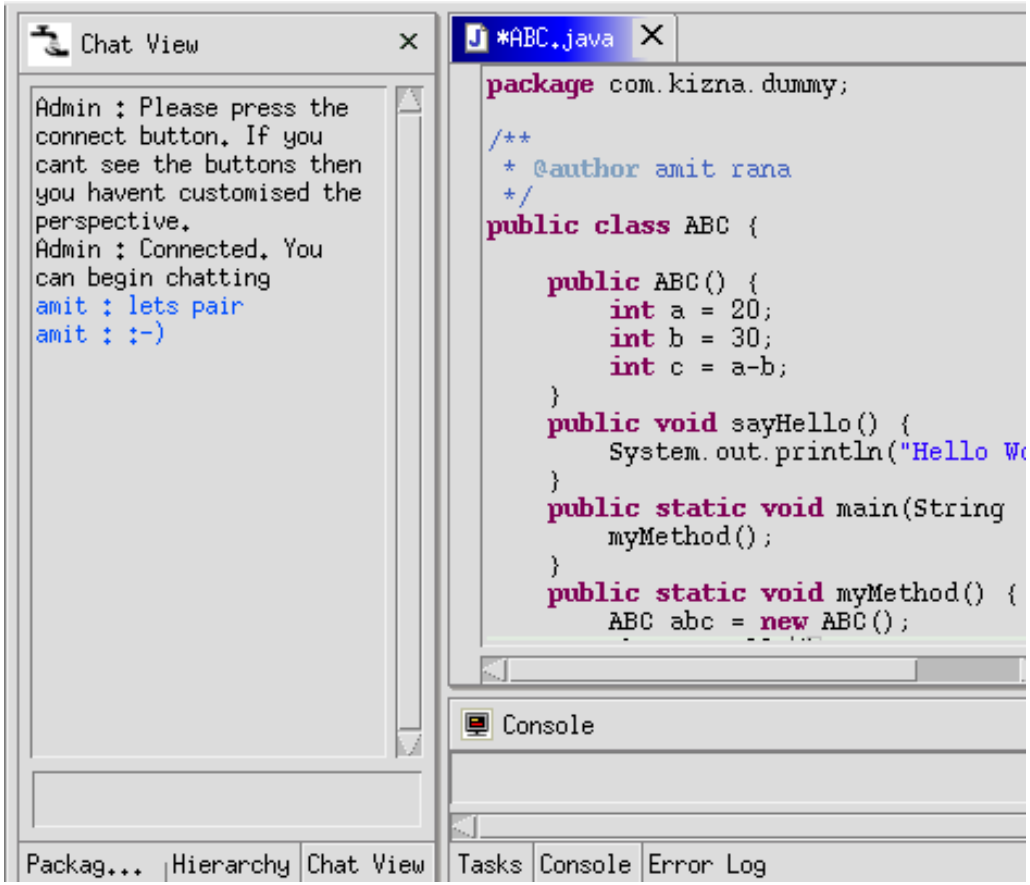
- PMD scans Java source code and looks for potential problems like:
  - Unused local variables
  - Empty catch blocks
  - Unused parameters
  - Empty 'if' statements
  - Duplicate import statements
  - Unused private methods
  - Classes which could be Singletons
  - Short/long variable and method names
- About 8 months on the Eclipse list

Sangam

- Pair programming (remote)
- 6 months on Eclipse list
- 8 months on Eclipse list

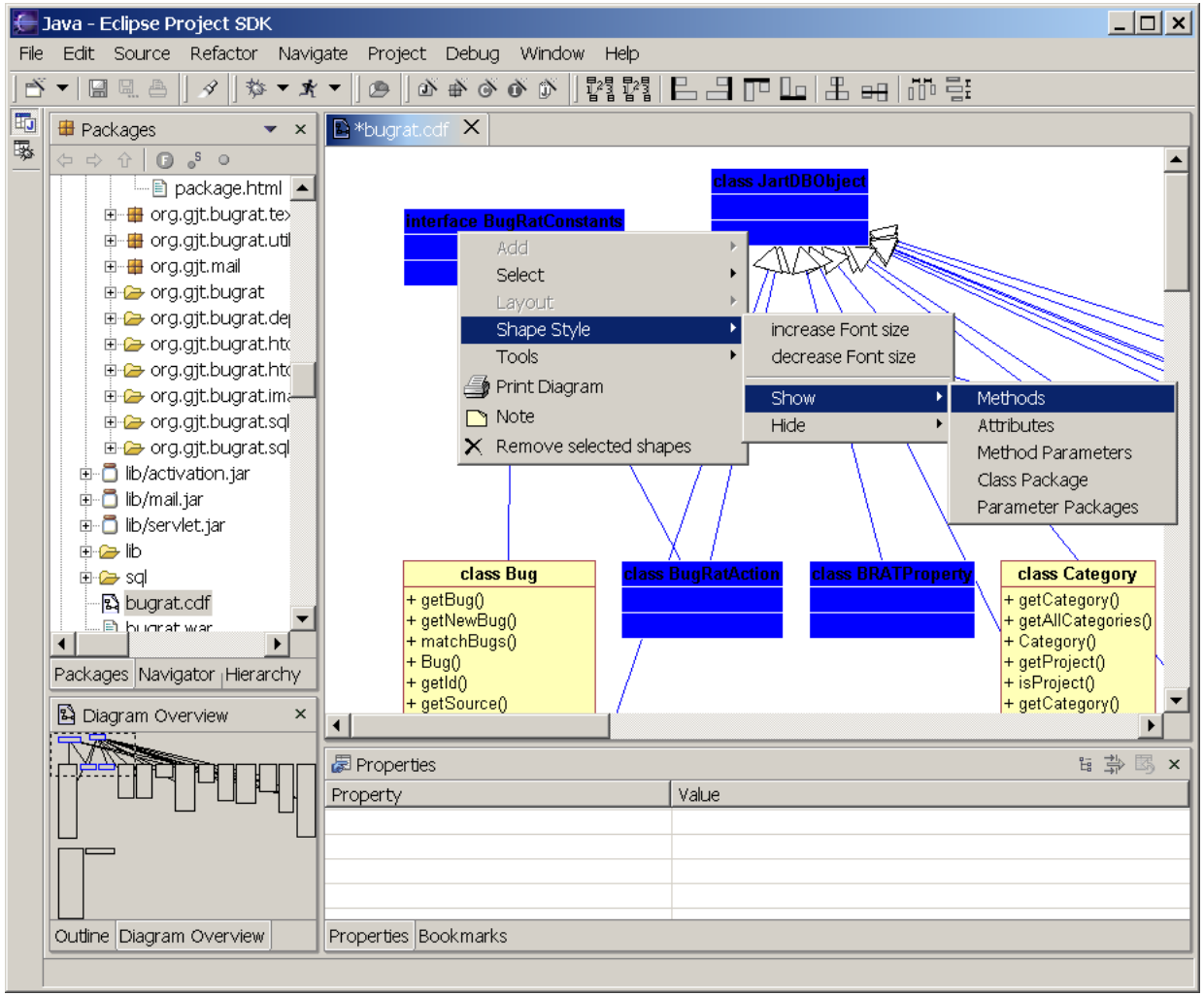






SlimeUML

- UML Diagram tool
- Updates as you type
- Commercial
- About 8 months



Others

- Annotations
- Code Formatters
- Web development
- Other languages
- SHriMP….
- and many more …

Long Term Vision....

- More courses (not just first year)
- Training
- Framework for developing a research core
- Convey research practices to other disciplines
- Infrastructure (organizational)
- Community (collaborations, workshops)

Next Steps

- curriculum to complement tools
  - tools shift how we design curriculum
  - dependency graph of concepts
- requirements gathering
  - curriculum
  - tools
- select research methodologies
- get involved with other projects