DrJava Development

Charles Reis
CSE 403 Guest Lecture
DrJava

- Pedagogic Java IDE
- Simple, Interactive
- Used at dozens of schools around the world
- Freely Available
Development Overview

• Written by students at Rice University
  • Graduate, Undergraduate
  • High rate of turnover

• Open Source

• Extreme Programming
Development Team

Professor

- PhD Student
- Masters Student
  - Undergrad
  - Undergrad
  - Undergrads
    - Comp 312
On the Inside

• Java with generics (List<String>)
• ~400 classes, 50,000 lines of code
• Complex (unit tests are critical!)
  • Two JVMs, plus multithreaded GUI
  • RMI, JDI, Custom Classloaders
  • Backward compatibility
Benefits of Open Source

- Freely available
- Tool and Management Support
- Incorporate existing code
- Educational value
- Word of mouth, Credibility?
Tools and Management

• Sourceforge.net
  • Free hosting for 80,000 projects
  • Professional management tools
    • Track features, bugs, tasks, support
  • Ant, JUnit, CVS
Use Existing Code

- **Dynamic Java**
  - Java source code interpreter
  - Critical to DrJava's quick maturity

- **JUnit Integration**
Educational Value

- Source code available for students, tinkerers
- Credible use of undergrads in Comp 312
- Building block for research tools (DrScala)
Complications

• Choice of License is tricky
  • **GPL**: true "free software"
    • All incorporated/derivative works GPL'd
  • **BSD**: more flexible, fewer guarantees
    • Allows us to use JUnit
Extreme Programming

• Simple practices that work well together
  • Pair Programming
  • Unit Testing
  • Continuous Refactoring
  • Incremental Development
  • On-site Customer
Typical Activity

• Prioritize bug reports
• Write test to exhibit bug
• Pair program to fix bug
• "Commit" (update, compile, test, commit)
• Release
Releases

• **Theory**: repository can always be released

• **Practice**: not exactly...

• Development releases (weekly/monthly)

• Stable releases (a few each year)
Life Cycle

- Peak development in spring and summer (Comp 312, summer interns)
  - 3-4 large features, many small fixes
  - Masters Theses
- Maintenance in "off-season"
Lessons Learned

• Unit tests are essential to stability
• Work incrementally
• XP is effective for high turnover
• Much to be gained from open source, even without many external developers
Difficulties

• Hard to test (and design) GUIs
  • Hard to enforce good test coverage

• Concurrency can be a mess

• Java isn't really platform independent...

• Tough to keep documentation up to date

• Maintenance/support is a full time job
Closing Thoughts

• Immensely satisfying to work on a widely used product

• Open source is a great fit for academia (perhaps elsewhere as well)

• XP can work very well for small teams
More Info

- http://drjava.org
- creis@cs.washington.edu