Experience in the Design and Evolution of ArgoUML and Tigris.org

Jason Robbins
UC Irvine
Introduction

• This presentation explores
  – Design changes in ArgoUML over time
  – Comparisons to other open source projects
  – Evolution of the Tigris.org community
  – Motivations for change in open source projects
  – Potential areas of improvement

• It raises questions for discussion rather than answering them
Some Previous Ideas on Design Evolution

- Spiral model: risk-driven change
- Seeding, evolution, reseeding
- Extreme programming
  - Release negotiations
  - Refactoring as normal development
- Open source development
  - Scratch an itch
  - Imitate leading products
  - Implement standards
ArgoUML Case Study

• Era 0: Seeding
  – A “lamp store” of cognitive features
  – 100 KLOC + one library
  – Standards: UML, Swing
  – Solaris-centric, java only, not i18n’d
  – No formal release process
  – CVS read-only, jobjar.html for issues
  – Deploy: source, .jar
ArgoUML Case Study

• Era 1: Building the community
  – Bug fixes
  – Better build process
  – Move to tigris.org: mailing lists, issue tracking, writable CVS
  – Architecture: core + plug-ins
  – Deploy: source, .jar, JavaWebStart
ArgoUML Case Study

• Era 2: Marko and Toby
  – Switch from custom code to standard libraries:
    • MM to NS-UML
    • Log4j
  – Internationalization (i18n)
  – Completed UML support
  – Commercial fork
  – Team structure: component owners
  – Removal of some research features
ArgoUML Case Study

• Era 3: Markus and Linus
  – 200 KLOC, plus more libraries
  – Plug-ins; multiple languages
  – Layers of abstraction around libraries
  – Deploy: multi-platform installers
  – Process improvements
    • Huge increase in documentation
    • Automated tests, nightly builds, docs
    • More regularly scheduled releases
    • Much more issue tracking (2250+)
ArgoUML Case Study

• Era 4: The next year
  – More spun-out components
  – Return of UI polish
  – Greater user support: A book
  – Limited new features
ArgoUML Design Aspects

• UCI sponsorship initially, then community driven
• UI: pretty stable, but i18n’d
• Critics: maintained
• Meta-model:
  – This is the core data structure
  – It has been replaced twice
  – Now adding level of abstraction around it
• Event propagation:
  – From Java propertyChangeEvent in MM
  – To EventPump mediator
Other Projects: GEF

- Imperfect spinout of ArgoUML
  - Duplicated code
  - Perceived as “part of ArgoUML”
  - Original design under-documented

- Only major contributions were dumped on project without follow-through
  - Uncertain quality of result; dead code
  - Package structure drifted

- Successful changes just mirror ArgoUML
  - e.g., i18n, new build processes

- Only a few projects reuse the GEF library
  - Only one major open source project uses it (ArgoUML)
  - Shortage of new contributors and requirements
  - Has not kept up with external standards (Java2D)
Other Projects: Subversion

- Well-scoped requirements and basic design choices from start
  - Developer-driven feature set
  - Multi-platform and i18n from start
  - Stayed with initial component choices
  - Stable, standard protocol: WebDAV
  - Stable project leadership
  - Stable build and test process
- Regular releases; happy users
- Many SVN clients by other people
Other Projects: Scarab

• Ambitious scope from start
  – Customer-driven laundry list
  – Tried to generalize in too many ways
    • I18n, DB neutral, servlet container independent, easy-to-reorganize UI
    • Hard to work with very indirect code
  – No particular standards to follow

• Sporadic releases; less happy users
  • Diversions into building new libraries
  • Deep bugs in assumptions that were never tested for feasibility
Evolution of Tigris.org

• Founded 3 years ago with ArgoUML, GEF, Subversion, and Scarab
  – Focus on building open source software engineering tools. And, developing and promoting CollabNet technologies.

• Now 20,000 registered users, and 200 active projects
  – SCM, issue tracking, requirements management, design, technical communications, testing, deployment, libraries, process, professional development, and student projects
Evolution of Tigris.org

- User views of Tigris.org
  - Conducting a survey now

- Future plans
  - Grow by inviting successful projects to move to Tigris.org, specifically testing and analysis tools
  - Provide more textual content on software engineering basics and trends
  - Bring featured projects to 1.0
Motivation for Design Changes

- Scratching an itch
- Completing the thought
- Broadening the user base
  - I18n, cross-platform, DB independence, easier installation
- Creeping internal quality
  - E.g., improved testability, use of latest libraries, improved performance or modularity
Other Factors in Change

- Initially ignored key concerns
- Strong project leader with clear roadmap for change
- Project membership continuity
- External evolution of relevant standards and libraries
- External evolution of commonly accepted development practices
Areas Ripe for Improvement

- Reduce need for design changes
  - Better initial requirements
  - Early feasibility evaluations
  - Start with more fully-formed application skeletons and frameworks, more stable libraries
- Continuity of team knowledge throughout project
- Help plan and carry-out changes
  - Help in managing product roadmap
  - Aspect-oriented programming or refactoring tools for a few key aspects