Can Extreme Programming Work?

Paul Wolgast
Paul.Wolgast@jpl.nasa.gov
x34998

Jet Propulsion Laboratory
• Highlights one experience applying some aspects of XP
• Presents few hard metrics to back up any assertions
• Not a silver bullet to make software development any easier
• The project where XP was applied originally was cancelled…
Agenda

- Project Goals
- The Project
- The Risks
- The XP Premise
- The Role of Tests
- Perspectives
Project Goals

- Provide an industry standard specification to:
  - Influence commercial directions in this area
  - Provide a focal point for our customer's efforts
- Develop a product that could:
  - Be the basis for other commercial products
  - Be the basis for a DISA product
  - Redistribution in the open source community
  - Run on any Java compliant platform with no recompilation
The Project

- Java Community Process (JCP) Project
- Builder SDK Features
  - Generates "zero-footprint" installer
  - Usable in GUI or batch toolkit contexts
  - Provides ability to save customization work for use later
  - Extensible using standard Java Beans
- Installer
  - Executable jar file runs on any platform that supports Java
  - Used in interactive GUI or batch contexts
  - Generates an executable uninstaller during install
- 365K lines of software in 1535 files including javadoc
  - 225K/140K code/comments ratio in 1290 files
  - 55K/15K code/comments ratio for unit tests in 245 files
- 2 products, JIFI Framework and the JBCI product
- Compatibility Test Suite and Specification produced as well
- Documentation set

June 25, 2003
The Installer Builder

GUI Builder

Custom Program

Installer / Builder SDK

Product Definition

Page Definition

Installer Runtime

Executable Installer Archive

June 25, 2003
The Installer Product

Welcome to the Installation of xyz
Press the Next button to proceed.

Executable Installer Archive

URI

Installer Runtime

Installed Bits

Registry

June 25, 2003
The Team

- ~5 local software developers
  - A mixture of skill levels
  - Some turnover
- 4-5 contributors from other organizations (Sun Microsystems, Informix)
  - 2 junior
  - 2 senior
  - Not always available
- Development spread from Portland, to San Jose, to Inglewood, to Pasadena

June 25, 2003
The Risks

- How to keep contributors (true volunteers) engaged
- How to meet expectations of our customer(s)
- How to “succeed”...

- Project ostensibly focused on specification development, however:
  - Customer really wanted a product
  - Customer wanted us to get Industry to build the product
  - Individual contributors really wanted to build a product
- “Successful” completion requires a specification, a reference implementation (the product) and a test suite
Project Timelines

2000 2001 2002 2003

JSR 38 Specification Work (completed Nov, 2002)

JIFI Development (completed Sep, 2002)

JBCI (completed Nov, 2002)
The XP Premise

Requirements Analysis Design Implementation Testing

Cost of Change

Time

June 25, 2003
The XP Tenets

- Iterative Planning
- Short Release Cycle
- Simple Design
- Testing
- Refactoring
- Pair Programming
- Collective Ownership
- Continuous Integration
- 40 hour Work Week
- On-site Customers
- Coding Standards

Legend:
Applied
Partially Applied
Not Applied

June 25, 2003
Planning

- Requirements collected at start from all members of JSR
- For JIFI, requirements came from original program
- Planning centered on frequent release approach
  - Integrated into development
  - Estimates never went beyond three weeks
  - Long range planning focused on broad release goals
- Customer involvement provided regularly through:
  - Direct involvement in iteration planning
  - Receipt of releases on frequent basis
  - Sun was most significant customer for JIFI product
  - For JBCI product, customer was DISA integration and test engineers

June 25, 2003
Development Process

- Design phases include the use of UML:
  - Use cases
  - Interaction/Sequence diagrams
  - Class diagrams

- There were many Design Phases, each kicked off an iteration start
- Most design sessions lasted three days and were off-site
- Iterations were on ~three week intervals
- Each iteration resulted in a completely integrated product
- Bugs tracked through DevTrack tool
- Bug ID’s used for all code checkins
- Coding standards “shaped” through JStyle tool

June 25, 2003
Development Process (cont)

- Daily discipline:
  - Concurrent versions system (CVS) used to administer source
  - Code checkins proceeded iff:
    - Source built cleanly
    - All unit tests run cleanly

- Code development:
  - Design a unit test first
  - (Rethink your original code design?)
  - Write your unit test
  - Write your code
  - Compile
  - Run your unit test against the code
  - Run all unit tests with your new changes
  - Check in code

June 25, 2003
Continuous Integration

• Continuous integration made possible by
  • Unit tests
  • System tests
  • Easy and very quick build system
  • Concurrent version control
• Each engineer assumed the role of ‘build master’ on two week cycle
  • Automated build sent email to the build master every morning
  • Build master monitored problems, fixed any that could be fixed, delegated remainder
• Each cycle resulted in a product that
  • For JIFI was released internally at Sun
  • For JBCI was released to test and integration engineers

June 25, 2003
Continuous Design

- Only those features targeted for the current iteration
- Features were selected by what was needed at the moment
- Other features can be added when they are targeted for release
- Unit tests facilitates ongoing change
- Refactoring solves the problem of later feature introductions
- Refactoring used to evolve the design
  - Reverses the “design, then code” order
  - The answer to ongoing design/code iteration methodology
  - Provides well known patterns for changing/improving code
  - Patterns described in Refactoring by Martin Fowler
- Examples:
  - RMI-based remote strategy designed and integrated distributed operations
  - Fileset unit refactored and integrated
  - Console unit rewritten
The Role of Tests

- We use the Junit test harness (www.junit.org)
- Unit tests provided for:
  - Any new functionality
  - Any corrected bug
  - Any new integration point
- Unit test provide a number of benefits
  - Developed first, provide excellent focus on the problem
  - Forces you to consider boundary cases explicitly
  - Provide a ‘safety net’ for related changes downstream
  - Helps others adding changes in other areas
  - ‘Documents’ features better than many comments could
  - Provides a regular metric on the health of the overall system
- 350 individual tests run nightly on the system
- System test capabilities also built in and used in nightly builds
The Role of Tests: An Example

- Halfway through the project we replaced the build system with Ant
  - Project must be able to build on a multitude of platforms including
    - Windows
    - Linux
    - Solaris
    - Other *nix variants including OS X
  - This change was low risk, (because of our test suite) AND:
    - Reduced our build time
    - Allowed us to ship source confident that anyone with a VM could build it
    - Aligned us with build tools with which the developer community is familiar
    - Allowed us to spread our nightly build/test cycles to more platforms
    - Was aligned with our goal of delivering a cross-platform tool
Management Perspective

- Liked the idea of unit tests
  - Knew the product was working and testing regularly
- For development, prefers iterative approach:
  - Multi-release plan
  - Provides better sense of what level project is operating
- More senior people needed:
  - Continuous integration
  - Continuous refactoring and design
Perspectives - Developer

- Thoroughly supported the unit test concept
- Joined the group in the middle of the development
  - Unit tests helped guide her through the initiation
  - Coding unit tests first always left her with a positive indication:
    - Focus you on what the development task was,
    - Clarified the API, made it make more sense,
    - protected you from false starts
- Wished there were better system tests as well
  - We should have changed the order we completed items
- Documentation was missing, wrong, or incomplete
- Pair programming was good for junior people but could be a real hindrance to senior people

June 25, 2003
Perspectives – Developer

- Liked the unit tests
  - “gave us more confidence in our system”
  - “could attack our problems better”
- Liked nightly builds
  - Good early detection system
- Short iteration cycles helped keep project focused and on track
  - Belief that this approach works well for new development
    - Projects incorporated new technologies
    - New product idea development
  - May not be good for maintenance since:
    - You can see further into the future with more confidence
    - Consequently you can have longer term plans
- Didn’t necessarily embrace XP software ownership model
  - “Pride of ownership” is lost
Perspectives - Developer

- Liked the unit tests, especially when approaching a new project
- Liked the daily builds
- Liked the continuous integration because it finds problems when they occur

"I want my unit tests!"
Do It Again?

- Unanimously approved for repeating in the future
- Customer involvement through all phases cannot be emphasized enough!
- The existence of unit tests is not enough
  - Need to include unit tests as part of the code reviews
  - Many of the unit tests did not test the right thing
- Documentation should be part of code reviews
- Many aspects applicable to a wider range of project types including:
  - The role of unit tests
  - Continuous integration and builds
  - Collective ownership
    - Great way to review other’s work
    - Great way to learn from other’s work

June 25, 2003
Other Influences

- **Dynamics of Software Development**, Jim McCarthy (Microsoft Press)
  - Multi-release technology plan
  - "If people trust in the future, they don't feel compelled to get everything done this time"
  - People see the next release as an opportunity to do the stuff they want
  - Produce a vision with everyone's involvement to gain trust and group cohesion

- Ben Rich's book: **Skunkworks**
  - A hardware-based approach that may provide lessons
    - Accomplishing innovation on a scale never seen before
    - Using technologies that are new in the development of new products
  - What I gleaned from their experience:
    - Teams must be small
    - Accommodations must be made up front for course changes
    - Design engineers collocated with fabrication engineers

June 25, 2003
Resources

- Junit: http://www.junit.org/index.htm
- Ant: http://ant.apache.org/
- Extreme Programming:  http://www.extremeprogramming.org/
- JIFI Installer/Builder downloads:  http://www.openinstallation.org
  - Open source product (Apache style licensing)
  - Full documentation
  - Full source and binary
  - Cross platform, explicit support for Windows NT, 2000, XP, Solaris, Linux

June 25, 2003
Backup Material
Other Software Experiences

- Unix O/S products
  - Device drivers
  - Kernel modules, network, system calls etc.
  - "Wild West" approach
- Animation Software project (~5 developers)
  - Suite of 6 graphical and computationally intensive applications
  - Unit test suite run nightly
  - Nightly builds
  - Resulted in very reliable and robust software
- Middleware software project (~8 developers)
  - Unit test suite run nightly
  - Nightly builds
  - Build master

June 25, 2003
Product Features

Installer Builder Features

- Generates "zero-footprint" installer
- Usable in GUI or batch toolkit contexts
- Provides ability to save customization work for use later
- Extensible using standard Java Beans

- Installer
  - Executable jar file runs on any platform that supports Java
  - Used in interactive GUI or batch contexts
  - Generates an executable uninstaller during install

June 25, 2003
Product Uses

- Deploying applications that run on different platforms
  - A single installer can support all targeted platforms
  - Each target-specific application is installed on the correct platform
  - Each application targets the appropriate machine through Rule/Action usage
- Deploying data that is platform independent
  - A single installer can target multiple platforms with no customizations
  - Installing web content or other forms of documentation
  - Installing java class files
- Use as a Process Manager
  - Provides simple graphical cues to user
  - Provides customizable flow controls
  - Provides customizable actions to allow user to initiate processes
Other Features

- Builder state can be saved in a file and reloaded later
- Saved files can be executed using the batch builder mode
- Installer can be run in silent (unattended) mode
- New actions and rules can be built and integrated into the system
- Open source implementation – Apache style license
  - Full source available with binaries
- Three customers currently using product
  - DISA's new installer based upon this technology
  - Reference implementation in use on Lab
  - Sun Microsystems's new installer development based upon this software