CSE 403: Software Engineering

Tools for Developing Software in Teams
Introduction

David Notkin (notkin@cs) Instructor
Kıvanç Muşlu (kivanc@cs) TA
Anton Osobov (aosobov@cs) TA

• This lecture aims to summarize the information available at Google Doc
• Please comment it with your knowledge and experience – requires @cs accounts
• PDF version is available on the web page
CSE 403 Tool Needs

• Produce documents and draw designs
• Implement code as a team
• Write unit tests, documentation, etc.
• Produce internal and external web pages

• What is the most efficient way to do these?
Phase 0: Project Hosting

• Online hosting sites are *highly* encouraged
  – Comes with version control system, bug tracking and wiki

• We suggest
  – Google Code: Quick Fix Scout and Google Web Toolkit
  – Github, Bitbucket or Sourceforge

• Private hosting at CSE servers is also possible
  – However, you need to convince us about why this is crucial for your project
Version Control Systems: Why?

• Allows developers to work concurrently
  – Personal changes do not effect each other unless published

• Preserves a history of the changes
  – If something goes bad developers can rollback to a ‘stable’ state

• Using a VCS is a natural requirement!
  – If not, believe us that you WILL regret it!
Version Control Systems: Types

• Distributed (suggested): Mercurial (HG), Git
  – A “central” repository
  – Distributed private (personal) repositories
  – Personal workspaces
  – Make changes => commit locally => push to share

• Central: CVS, SVN (Subversion)
  – A central repository
  – Personal workspaces
  – Make changes => commit to share
Phase 1: Design

• Quite complex ranging from:
  – Low level decisions (e.g., which data structure to use)
  – High level abstractions (e.g., architecture diagrams)

• In any stage, Unified Modeling Language (UML) is very popular to generate diagrams
  – e.g., Omondo plug-in for Eclipse
  – There are lots more on the web page!
Omondo UML Example
Phase 2: Implementation

• Choose a programming language
  – High level languages (e.g., Java, C#) are suggested

• Choose an IDE: Eclipse, Visual Studio, etc.

• Free to choose the language, platform, IDE.
  Just make sure that your final implementation
  – Conforms to software engineering standards
  – Consistent with your requirements, design & specifications
Phase 3: Testing

• Finds bugs in your implementation
• Checks its correctness (up to a degree)
• Available Frameworks
  – JUnit (Java, comes with Eclipse)
  – .Net includes testing for C# (comes with VS)
• Quality vs. quantity of the tests
  – Coverage tools to approximate quality: Java
Extra Tools

• **Crystal** (contact Yuriy Brun)
  – Works with Git & Mercurial
  – Notifies developers as soon as conflicts occur
  – Can save a lot of time!

• **Quick Fix Scout** (contact Kıvanç Muşlu)
  – Works with Eclipse Classic 3.7.1
  – Helps you to resolve compilation errors easier

• They are **NOT** required but can help you
  – We would appreciate the feedback
Initial Advice

• Quarter is short, don’t stuck with details
  – Try to keep up with the internal deadlines
• If you are unsure about something, ask us
  – We don’t also know every available thing out there, but will do our best to help!
• If you ever have a problem about the course requirements or your team, let us know
Final Words

• Learn available tools and technologies
• Learn how to build software as a team
• Have fun through the quarter!
Reminders

- **Webpage**: Updates & announcements
- **GoPost Forums**: Discussions & lecture summaries
- 3+ people in each group (one must be a program manager, a “PM”)
- Project proposals descriptions and slides (presented by you at lecture on Friday)
  - Proposals based on the KNOW (Jackson School) ideas or on your own ideas
- Thursday, we meet again to form the teams
- Project and team preferences submitted by Friday night
- Projects and teams announced by Saturday night