Professional Development Environment

Lecture on

Professional Development: Tools and Processes

An Introduction

Walter Kriha

Goals

- Learn to master the extensible open source IDE eclipse. Use its plug-in concept to integrate your own functions
- Learn tools needed for large scale development: Source code control, automatic build, unit tests etc.
- Integrate those tools into a flexible development environment
- Learn the processes behind large scale development: heartbeats, release management
- Learn the "soft factors" behind software teams and processes

In our exercises we will create a live environment comprising the essential tools and utilities needed for large scale development.

Non-Goals

- This is NOT a modelling class. We will possibly generate some source but the focus is definitely not on UML or other modelling tools.
- This is NOT a software architecture class. Architecture does have effects on a development environment (packaging, work distribution etc.) but we will not learn how to build frameworks.
- This is NOT a project management class. It tackles team development strictly as a technical issue. But it favors "agile" methods like extreme programming.

At the end of this class you should have hands-on experience with many development related tools and you should understand the process of developing software in teams. Development tools change frequently but the main processes and forces stay with us.

Procedure

We will create our environment in much the same style as it would be the case in a real software team:

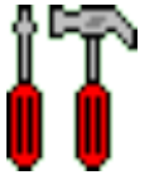
-members will take over responsibility for tools or processes, learn those and teach their use to the rest of the team.

-the team will discuss all configuration issues (like source structures, coding standards etc.) and define processes.

-at the end we will have a complete, integrated environment and a process to create a software product.

For every topic a short introduction will be given (one hour) and then we go to the lab.

For every tool:



We define the forces behind it and why we use the tools

We define how the tool interacts with others and how it is integrated

We show how it is installed and how it works

We list best practices and shortcomings.

A tool only makes sense when it is integrated into a development environment and a development process

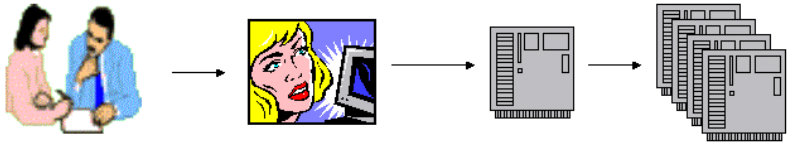
Development Process Overview

- architectural artifacts
- customization plans
- lifecycle plans
- collaboration support

- coding standards
- automatic builds
- source code control
- release management
- source structure
- packaging
- unit tests
- micro tools
- Compile tools

- test configurations
- load tests

- deployment configuration
- per client customization
- bug reporting



Many different forces exist for a development environment and process: architectural, life-cycle issues, customization, collaboration etc. (see Kriha et.al. in resources) The most critical tasks are those that define the future extensibility of the product. Will it be a single application? A toolkit or framework? A production line? Other important questions are whether you will need to setup a distributed development team and how to organize a development/test/production zone.

Tool mapping

- Modelling tools´(use cases etc.)
- WikiWiki

- coding standard
- ANT/make
- CVS
- WikiWiki
- Junit/Httpunit
- homegrown tools
- IDE with debugger
- Memory tracer
- bugzilla
- documentation editor
- batch processor

- loadrunner
- optimize it
- jprobe
- WikiWiki

- bugzilla

The tool chain is intimidating but necessary. Everyone in a software team needs to take responsibility for some tool(s). Be careful not to pick tools which will not fit to the size of your project: If you use Clearcase instead of a simple CVS you will need an extra administrator. Is at least one team-member familiar with a complicated IDE to give you a headstart? Do the tools support the number of developers? How long will it take to make everybody familiar with those tools?

Resources (1)

- www.eclipse.org home of the eclipse project
- www.sourceforge.com portal for open source projects
- www.apache.org home of the ant automatic build tool.
- kriha et.al. on large scale structures: http://www.kriha.de/krihaorg/docs/papers/largesystems/LargeSystems.html