# Design of Software Systems (Ontwerp van SoftwareSystemen)

7 On Multi-user Development Tools, Versioning and Packaging of code, their Relations, the Universe and Everything.

> Roel Wuyts 2015-2016

# Developing Complex Systems

How do scientific disciplines construct complex systems ?

# Architectural Software



Roel Wuyts – Design of Software Systems Creative Commons License 4

# RF/mW Design & Analog/RFIC Verification



Roel Wuyts – Design of Software Systems Creative Commons License 4

# Visualization & Manipulation of molecules



# Computer Science/Engineering...

Buffers Files Tools Edit Search Help = AL/3600.0E+0 AL SPA = DPA Ĥ = 1.0- 4.6747D-5 в = A\*\*3/6.0/206265.0E+0\*\*2 PARA = SPA\*(A+B\*SPA\*SPA)/3600.0E+0 = T / 36525.0E+0 Т UTL = (( -17.2327E+0 + .01737E+0 \* T)\*SIN(OM) + ( -1.2729E+0 -0.00013E+0 \*T)\* SIN(2 \*OM + 2\*F - 2 \* DD) + ( .2088E+0 + .00002E+0 \* T) \* SIN( 2\*OM) + ( .2037E+0 + .00002E+0 \* T) \* SIN(2\* OM +2 \* F))/ 3600 OL = OL+UTL OL = AMOD(OL, 360.0E+0) UTE = (( 9.21E+0 + .00091E+0 \* T) \* COS(OM) +( .5522E+0 - .00029E+0 \* T)\* COS(2 \*OM +2\*F -2 \* DD) +( .0909E+0 + .00004E+0 \* T) \* COS(2\* OM) +( .0884E+0 - .00005E+0 \* T) \*COS(2\*OM+2\*F))/ 3600 = 23.0E+0 + 27.0E+0/60.0E+0 +8.26E+0/3600.0E+0 -46.845E+0\*T/3600.0E+0 - .0059E+0\*T\*T/3600.0E+0 + .00181E+0 \* T \* T \* T / 3600.0E+0 Е = E+UTE = SIN(AL \* DTORAD) SB CB = COS(AL \* DTORAD) SE = SIN( E \* DTORAD) CE = COS( E \* DTORAD) SL = SIN(OL \* DTORAD) Ĥ. = CB \* COS(OL \* DTORAD) в = CB \* SL \* CE - SB \*SE = CB \* SL \* SE +SB \* CE CC DELTAM =ATAN2(CC,SQRT(1.-CC\*\*2))\*RTODEG BPERA = B/ABPERA = BPERA/SQRT(1+BPERA\*BPERA) ALFA1 = ATAN2(BPERA, SQRT(1.E+0-BPERA\*\*2)) IF (A .LT. 0.0) ALFA1=ALFA1+PI IF (A .GT. 0.0 .AND. B .GT. 0.0) ALFA1=ALFA1 +PI2 ALFAM = ALFA1\*RTODEG/15.0 IF (ALFAM .GT. 24.0) ALFAM=ALFAM-24 IF (ALFAM .LT. 0.0 ) ALFAM=ALFAM+24 RETURN ----Emacs: nb.f Garbage collecting...done

# Corollary

We need to construct systems that are typically more complex than in other disciplines

- for several reasons

### We have tangible elements to manipulate

 Buildings, circuits and molecules need a representation that is different than their physical one

- Yet lots of developers still seem to prefer basic tools
  - yes, emacs is a basic tool...

# Eclipse/Netbeans/IntelliJ/...?

🖨 Java - Task 99 - Eclipse SDK		
Ele Edit Navigate Search Project TeamPlain Run Wir	ndow Help	
] 📑 • 🔜 🗁 ] Go to Work Item ] 🏇 • 💽 • 💁 •	] 🖄 🛱 ⓒ • ] 🥭 🔗 ] 🏷 ⇔ • ⇔ • 🗄 🛃 Java	
🦉 TeamPlain Explorer 🗙 📃 🗆	All Work Items [Results] 🔚 Task 99 🖄 🛅 Bug 117 🙋 Active Bugs [Query]	- 8
▽ ⇔ 令 🚮 凶 🖻	(	Э
<ul> <li>http://vstsb3r:80800</li> <li>My Favorites</li> <li>Adventure Works (CMMI)</li> <li>Work Items</li> <li>All Way Team Project Work Items</li> <li>All Work Items</li> <li>My Work Items</li> <li>Resolved Bugs</li> <li>My Queries</li> <li>Contoso (Scrum)</li> <li>Northwind</li> <li>Work Items</li> <li>All Quality of Service Requirements</li> <li>All Scenarios</li> <li>All Work Items</li> <li>All Work Items</li> <li>All Work Items</li> <li>My Work Items</li> <li>My Work Items</li> <li>All Quality of Service Requirements</li> <li>All Scenarios</li> <li>All Work Items</li> <li>My Queries</li> <li>My Cueries</li> <li>My Queries</li> <li>My Queries</li> <li>My Cueries</li> <li>My Cueries</li></ul>	Task #99: Setup: Migration of Work Items         Title:       Setup: Migration of Work Items         Classification         Area:       Northwind         Iteration:       Northwind         Iteration:       Northwind\Version 2.0 Beta 1         Status       Assigned         Assigned       John         Vo:       Reason:         New       Reason:         New       Network Items         Iteration:       History         Links       File Attachments         Details       Field         New Value       Title         State       Active         Rew       1         Issue       No	
Problems Javadoc Declaration 🔮 Error Log 🛛	, I I I I I I I I I I I I I I I I I I I	8 🗋
3		

Roel Wuyts – Design of Software Systems Creative Commons License 4

# Eclipse...

Eclipse is a decent integrated development environment

- integrates navigation, editing, unit tests, refactoring, ...
- was developed by a lot of former Smalltalk people :-)

But at its core it is file-based (and so are most others)

- So ? Why don't I like this ?

# Files versus Objects

### Non computer science disciplines:

- Architects work with construction materials&buildings
  - So do their tools
- Molecular biologists work with modules
  - Environment manipulates molecules
- ...
- We work with objects
  - Most tools deal with files ?!

# Smalltalk image approach

# The Smalltalk image is a live environment

- consists entirely of objects
- objects are manipulated

# Files are one way of storing objects

- code too, since code are objects
- Databases are another mechanism, or network sockets or ...

# Sidenote on Environments

## Good developers tailor their environment

- So they need to be easily extensible
  - emacs: easy
  - Smalltalk environments: easy
  - Eclipse: possible
  - Most environments: hard or not possible

# Always favour an extensible one

- control your tools!

# Multi-user Development

Software engineering is a teamsport ;-)

Needed

- a code repository that allows multiple users
- integrated versioning
- configuration management
- The language also has packaging mechanisms
  - with or without namespaces
- These concepts cross-cut

# Code repositories and multiple users

## Need to store code (obviously)

- but preferable also binaries, documentation, tests, ...

## Locking vs. concurrent

- Lock: one user has (part of) code, unlocks when done
- Concurrent (lazy locking): several users can work simultaneously on the same system

### Centralized vs. Distributed

- Centralized: Only the master repository contains complete version history
- Distributed: all repositories have complete history

# Support for merging



Two-way merge

## Three-way merge





Roel Wuyts – Design of Software Systems Creative Commons License 4

# Example

One framework,

- instantiated for two different clients,
- each with their own customizations,
- Where there is a stable version,
- and two development branches
  - a new version and a brand new one
  - one dependent on the customization of the framework for one particular client

# Let's view two systems

Concurrent, centralized systems:

- Subversion (svn)

– Envy

Concurrent, distributed system:

– git

(Many more exist, but svn is archetypical for most popular tools like cvs/git, and Envy is a contrast)

## svn : Subversion

descendant of cvs (concurrent versioning system)

Granularity: file

Users work detached from the repository:

- Load local copy of files from svn server (repository)
- Work on local copy (working directory)
- Commit changed files back to repository

Loading local copy can be done from the network

# svn Workflow

Check-out code from repository in local environment

Work on code.

- can at all time see the difference between the current change and the state when checked-out
- When finished, commit changes back to repository
  - can trigger (3-way) merge when repository was updated in the meantime

## Semantics

## svn (like cvs, git, ...) versions text

- has no semantics about what text it stores
- works with latex files, C++ files, ...

#### Therefore its operations have no semantics

- e.g. looking at changes after doing a renaming a method refactoring result in a list of textual changes to potentially many files
  - can be hard to know it was a refactoring, especially when combined when several other changes
  - commit often, and add comments !

# Envy

# It's a versioning system, but not as you know it ;-)

- Users are meant to be always connected to the repository
  - can work separately but that is the exception
- Works with methods, classes, ...
  - Versioning knows about your language concepts
    - e.g. have all versions for a particular class, automatically includes all methods for that version of the class
  - Smallest granularity is a method

# Envy Workflow

Load code from repository in local environment

Work on code.

- Every change of a method or a class automatically (!) creates an edition
- These editions can be compared with, restored, ...

# Editions can be versioned

- the edition then gets a name and version number
- once versioned everybody in the repository can see and load versions
  - easier and earlier integration and conflict detection

# Envy: Configuration Management features

## From the ground up Envy has support for configuration management

- Applications group classes and methods
  - can have editions and versions themselves
  - have prerequisite versions (!)
- Configurations group applications
  - (e.g. Manifests in Microsoft .Net)
- Support for conditional loading and prerequisites
  - Platform-specific code, for example
  - Can be at application or configuration level

Removes need for external build systems like cmake, Maven, ...

## Distributed version control system

- Breaks the master/slave relationship prevalent in cvs/svn
  - every repository has the complete history
  - repositories sync with each other
- Good support for branching and advanced forms of version management (cherry picking, reverting changes, ...)
- Like svn/cvs/...: stores text

# Git workflow

Many possibilities

# Can of course do the centralised workflow (as in centralised approaches like svn/Envy/...)



Roel Wuyts – Design of Software Systems Creative Commons License 4

# Git workflow



#### Integration-Manager Workflow



Dictator and Lieutenants Workflow

# On granularity...

With svn/git/..., you have a <u>history of the files</u> you've checked in With Envy, you have a <u>history of the development</u> you did

This is fundamentally different !

# What is Envy doing in Eclipse ?!



Roel Wuyts – Design of Software Systems Creative Commons License 4

# Concepts in Code Repositories

#### Code

## Package

Configuration

# Packages and Namespaces should be orthogonal

- package contains definitions
- namespaces is a visibility mechanism

# Versioning

All the elements need to be versionable

Decisions, decisions:

- granularity of version
  - line of code, method, class+methods, package, ...
- forms of version numbers
  - single number, composed number, alphanumeric
- version numbers versus release numbers
  - and their relationships

# Concrete example : Menu Framework





# C++ Files



#### Files can go in cvs

- But decomposition is not the right one
- What if the visitor traversal needs to be changed?

# Java Packages



Packages to regroup classes, storage still in files

Decomposition still not the right one

- What would be the right decomposition?

# Smalltalk class extensions



Packages defines classes and/or methods

 Can be different versions, under control of different people/project/companies

## Note: declarative packages

Package systems should support software engineering and design principles

– e.g. packaging Visitor pattern

Approaches exist but should become mainstream

- Smalltalk's class extensions
- C# Partial classes and extension methods
- Java Open Classes (for example in MultiJava)

PS: or multi-methods in Lisp (and from there other languages)

- ...

# Software-engineering wise

Important to be able to separate development into logical, manageable pieces

- e.g. Visitor design pattern
- Each piece should have:
  - owners & responsibles
  - versions
  - dependencies
  - post-load and pre-unload statements

# Corollary

Good packages support evolution

- Company can sell parsetree
- Other company can sell visitor for parsetree

Code repositories and packages should support flexible forms of packaging code

Code repositories, packaging & storage are linked

## Sidenote

Design question:

- why is the plug-in mechanism in Eclipse so difficult?

## Last but not least

We discussed granularity

- want to see the development you really did, not the changes you made

Nice example: Refactoring Scripts in Eclipse

- Record and replay the refactorings you did

Why is this practical ?

# Saving & Replaying Refactoring Scripts



# Conclusion

## Need for supporting Multi-user development

- code repositories with concurrent access
- version support
- (automatic) merge support
- configuration management
- Current systems are quite weak
  - svn/git/... & files
  - proper packaging mechanisms
  - watch out for newer offerings

#### References

Subversion: https://subversion.apache.org/

git: <u>http://git-scm.com/book</u>

Envy overview: <u>http://stephane.ducasse.free.fr/FreeBooks/ByExample/36%20-</u> <u>%20Chapter%2034%20-%20ENVY.pdf</u>

Envy: Joseph Pelrine, Alan Knight, Adrian Cho, Mastering ENVY/Developer, Cambridge University Press, 2001.

Smalltalk Class Extensions:

https://www.youtube.com/watch?v=VNi\_VQMosXQ

# License: Creative Commons 4.0

#### You are free to:

- Share copy and redistribute the material in any medium or format
- Adapt remix, transform, and build upon the material

for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

#### Under the following terms:



Attribution — You must give <u>appropriate credit</u>, provide a link to the license, and <u>indicate</u> <u>if changes were made</u>. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.



**ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the **<u>same license</u>** as the original.

**No additional restrictions** — You may not apply legal terms or **technological measures** that legally restrict others from doing anything the license permits.