OOSCM
Object Oriented SCM

SCM Next Generation
We’d like to declare...

Barcelona
The SCM Conference
Why?

- Colin Putney’s Monticello 2
  - “Merging is the most important feature of an SCM”
- Veronica Urquillaz-Gomez
  - Torch: Code review before integration
- Fernando Olivero
  - Reify the programmer and his “unit of work”
- Dale Henrichs and Mariano Martinez Peck
  - Metacello
- And at the same time as this talk Alan Knigth
  - Taking about Store
We **NEED** a **BETTER** SCM tool
What are we going to talk about

OOSCM

- A new SCM concept
- WHY?
  - Current tools do not fulfill our goals
  - Current tools have conceptually different approaches to solve SCM that our idea
  - We want to promote Smalltalk
Motivation 1

- Where a change was integrated
- By who and when
- On what versions
- What were the changes made to that code during integration
- Provide feedback to the programmer at all levels (method, class, etc.)
Motivation 2

- Did all the tests run after implementing that change?
  - The programmer should run all tests before submitting the change to integration.
- Does the change provide new tests?
Motivation 3

▶ We don’t want this to be an integration conflict

Integration Conflict!
Motivation 4

- I want the decisions I made during the pre-integration code review to be applied during integration
  - Reformat the code
  - Do not integrate this method
Motivation 5

Baseline

1.0 Measure
- unit
- amount
- asMeters

1.1 DistanceMeasure
- unit
- amount
- asMeters

1.2 DistanceMeasure
- unit
- amount
- asMeters
- convertTo: aDistanceUnit

class renamed
method added

AUTOMATIC REFACTORYING INTEGRATION
Seaside releases version 3.1 that includes lots of method renames
- I want the system to tell me there is a new version
- I decided Patagonia should work with seaside 3.1!!
  - I can’t because I have to do the rename myself in Patagonia!
  - Why not applying those renames to Patagonia automatically?
Motivation 7

How do I know if programmers are using TDD or just testing?
- Writing the test first really

How do I know the system architecture and its evolution?
- Maven in Java, but cannot see its evolution or internal architecture, see dependencies.
What can we do with the current tools?
## Traditional SCM

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Traditional SCM (SVN, Git, etc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MANAGE CHANGE LIFECYCLE</td>
<td>NO</td>
</tr>
<tr>
<td>CHANGE QUALITY</td>
<td>NO</td>
</tr>
<tr>
<td>SMART AUTOMATIC INTEGRATION</td>
<td>NO</td>
</tr>
<tr>
<td>PRE-INTEGRATION TOOLS</td>
<td>NO</td>
</tr>
<tr>
<td>AUTOMATIC REFACTORING INTEGRATION</td>
<td>NO</td>
</tr>
<tr>
<td>AUTOMATIC UPGRADE</td>
<td>NO</td>
</tr>
<tr>
<td>QUERY THE REPOSITORY</td>
<td>NO (there is no model)</td>
</tr>
</tbody>
</table>
Traditional SCM

▶ Archive oriented: Archive is the minimum trackeable item
  – Between commit changes are lost! (lost of information)
  – Not easy to track group of changes
▶ No reification of what a module is, what a system is, etc. Just files and directories
▶ Not easy to model the development process
  – It has to be defined from outside the tool
▶ Good for versioning files, not so for OO development
▶ Advantages:
  – Simple interface
  – Can work offline
## Monticello/Metacello

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Monticello/Metacello</th>
</tr>
</thead>
<tbody>
<tr>
<td>MANAGE CHANGE LIFECYCLE</td>
<td>NO</td>
</tr>
<tr>
<td>CHANGE QUALITY</td>
<td>NO</td>
</tr>
<tr>
<td>SMART AUTOMATIC INTEGRATION</td>
<td>Medium</td>
</tr>
<tr>
<td>PRE-INTEGRATION TOOLS</td>
<td>Torch</td>
</tr>
<tr>
<td>AUTOMATIC REFACTORING INTEGRATION</td>
<td>NO</td>
</tr>
<tr>
<td>AUTOMATIC UPGRADE</td>
<td>NO</td>
</tr>
<tr>
<td>QUERY THE REPOSITORY</td>
<td>Yes (But limited to the info it stores)</td>
</tr>
</tbody>
</table>
Monticello / Metacello

- As traditional SCM, only "commits" are saved
  - Between commit changes are lost
- Metacello helps to model the system architecture
- But it does not help to
  - model the development group
  - Integrate different development lines
  - See the integration history (evolution)
- Advantages:
  - Simple
  - Can be use offline
  - Metacello provides tools that allow to use it easily
**Store**

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>MANAGE CHANGE LIFECYCLE</td>
<td>NO</td>
</tr>
<tr>
<td>CHANGE QUALITY</td>
<td>NO (could be added)</td>
</tr>
<tr>
<td>SMART AUTOMATIC INTEGRATION</td>
<td>Not sure</td>
</tr>
<tr>
<td>PRE-INTEGRATION TOOLS</td>
<td>Medium</td>
</tr>
<tr>
<td>AUTOMATIC REFACTORING INTEGRATION</td>
<td>NO</td>
</tr>
<tr>
<td>AUTOMATIC UPGRADE</td>
<td>NO</td>
</tr>
<tr>
<td>QUERY THE REPOSITORY</td>
<td>Yes (But limited to the info it stores)</td>
</tr>
</tbody>
</table>

* We do not have experience with Store, so this could be wrong.
**Requirements** | **ENVY**
--- | ---
MANAGE CHANGE LIFECYCLE | NO
CHANGE QUALITY | NO (could be added, in fact we did it)
SMART AUTOMATIC INTEGRATION | NO (we did it)
PRE-INTEGRATION TOOLS | Three way Differences
AUTOMATIC REFACTOING INTEGRATION | NO
AUTOMATIC UPGRADE | NO
QUERY THE REPOSITORY | Yes *

- New objects can be added to the repository
Trackable items are methods, classes, modules (application) and systems (configuration maps)
Between commit (versioning) changes are saved
Easy to see method, class and module history
Not so easy to see the system history
Kind of difficult for agile development (unless you remove security, etc)
No reification of programmers changes
No automatic integration (unless you develop it)
ENVY

Disadvantages:
- Proprietary
- Old server technology
- Complex implementation
- Only for Smalltalk

The best one we have used so far
OOSCM

Proof of concept
Manage change lifecycle
Manage change lifecycle
Manage change lifecycle
ChangeQuality

100.0 % Run - 800 run, 800 passed, 0 failed, 0 errors (0 to run, 0.0 %)

- Pass -

<table>
<thead>
<tr>
<th>Method</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>FloatTest</td>
<td>N/A</td>
</tr>
<tr>
<td>FractionTest</td>
<td>N/A</td>
</tr>
<tr>
<td>IntegerTest</td>
<td>N/A</td>
</tr>
<tr>
<td>LargeIntegerTest</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPBaseUnitTest</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPCanNotConvertMeasurementExceptionTest</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPConversionTableTest</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPDividedUnitTest</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPInvalidProportionalUnitConversionFunctionExceptionTest</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPMeasurementBagFormatterTest</td>
<td>5</td>
</tr>
<tr>
<td>MCPMeasurementBagFractionTest</td>
<td>6</td>
</tr>
</tbody>
</table>

All 800 passed, 0 failure(s), 0 error(s) out of 800 test(s)
ChangeQuality
## ChangeQuality

### Test Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Correct</th>
<th>Failures</th>
<th>Errors</th>
<th>Not Run</th>
<th>Total</th>
<th>Time (ms)</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>FloatTest</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>FractionTest</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>31</td>
<td>N/A</td>
</tr>
<tr>
<td>IntegerTest</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>LargeIntegerTest</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPEscUnitTest</td>
<td>47</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>47</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>MPCanNotConvertMeasurementExceptionTest</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPConversionTableTest</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPDividedUnitTest</td>
<td>37</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>37</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPInvalidProportionalUnitConversionFunctionExceptionTest</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPMeasurementBagFormatterTest</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>MCPMeasurementBagFractionTest</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Total:** 800 passed, 0 failures, 0 errors out of 800 tests.
No se puede proseguir con la integración automática porque: La aplicación GlorpDBTests DESA-FUN 2010-09-1 - 0.1 no está cargada.

Comando de integración del Configuration Map de Programador: MCP - Prog - Gabriel Cotelli DESA-FUN 2010-09-1 - 0.3
Comando de descripción de estado inicial del integrador
Comando de chequeo de la existencia de ediciones abiertas de aplicación que estén incluidas en MCP - Prog - Gabriel Cotelli DESA-FUN 2010-09-1 - 0.3
Comando de Integración de Configuration Map (MCP - Prog - Gabriel Cotelli DESA-FUN 2010-09-1 - 0.3)
Comando de Integración de Aplicación GlorpDBTests DESA-FUN 2010-09-1 - 0.1
Control de Aplicación GlorpDBTests tenga alguna edición cargada en la imagen.
SmartAutomaticIntegration
OOSCM

The solution...
OOSCM Goals

- SCM oriented to development with Objects
  - Paradigm shift from traditional SCM
- Easy to track programmers work
- Easy to track system evolution
  - Be able to model system architecture
- Automatic Integration
  - Smart
  - Easy to integrate complex changes (i.e. refactorings)
- Upgrades
  - Automatic upgrading
- Not only for Smalltalk:
  - Planned: Java with Eclipse
  - Open to other languages
Architecture

- Client-Server
- REST
- Multi-repository
- Offline support
- IDE Client: Plug-In Architecture
- WebBrowser support
Details

▶ Technology:
  – Developed with Pharo
  – Production: GLASS

▶ Project:
  – Subsidy of the Argentine Ministry of Technology (USD 38 K)
  – Total time: 13 months (with out refactoring integration)
  – We just started

▶ License:
  – Not sure, open client, close server?
  – Open for Smalltalk, paid for other languages?
  – Based on projects, customers, SaaS?
What do you think?
Thank you!

info@10pines.com
www.10Pines.com
twitter: @10Pines

Hernan Wilkinson
hernan.wilkinson@10pines.com

Jorge Silva
jorge.silva@10pines.com