Software Evolution from the Field: An Experience Report from the Squeak Maintainers

Marcus Denker
SCG University of Berne
Switzerland

Stéphane Ducasse
LISTIC
Université de Savoie
France

© Marcus Denker
Roadmap

> A little bit about Squeak
> Evolution problems
> Towards a solution
> — Software Engineering
> — Process
> — Programming Language
Squeak: Open Source Smalltalk

> Based on original Smalltalk
  — parts of the codebase are 30 years old

> Squeak is no toy! (even if it looks like one...)
  — 1600 Classes, 32,000 Methods

> We have been responsible for 3.7 and 3.9
Squeak: Features

> Two graphical user interface frameworks
  — MVC, Morphic
> Complete IDE with all tools
  — Incremental compiler, debugger, code browser...
> Language core and libraries
> eToy: programming for kids
> Multimedia support: pictures / movies / sound
> Various libraries: compression, encryption, networking
Communities and Projects

- Seaside: web framework
- Croquet: multiuser 3D
- Tweak / Sophie: media authoring
- SmallLand: Squeak for Kids
- SqueakLand: spanish schools (>40.000 PCs)
- Research (e.g. SCG Bern)
Squeak Development Process

> Up to 3.4: Alan Kay’s group
> Since 2001: real open source project
  — Squeak Foundation Board: Elected 2006
> For each release: maintainers
  — 3.9: Stephane Ducasse, Marcus Denker
> Release team (maintainers):
  — Integration (core)
  — Coordination (packages)
# Measurable facts

<table>
<thead>
<tr>
<th></th>
<th>year</th>
<th>#classes</th>
<th>#method</th>
<th>LOC</th>
<th>#changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>2003</td>
<td>1811</td>
<td>41408</td>
<td>322k</td>
<td>10</td>
</tr>
<tr>
<td>3.6</td>
<td>2003</td>
<td>1338</td>
<td>33277</td>
<td>246k</td>
<td>240</td>
</tr>
<tr>
<td>3.7</td>
<td>2004</td>
<td>1544</td>
<td>35526</td>
<td>261k</td>
<td>560</td>
</tr>
<tr>
<td>3.8</td>
<td>2005</td>
<td>1659</td>
<td>37953</td>
<td>281k</td>
<td>600</td>
</tr>
<tr>
<td>3.9a</td>
<td>2006</td>
<td>&gt;2000</td>
<td>&gt;44000</td>
<td>&gt;300k</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Common Problems

> Tangled code

> Dead code (ca. 30 years old!)

> Prototype code / old experiments

> Evolution dilemma:
  — How to provide a stable base and move forward?
Egocentric Syndrome

> Change means:
  — some bugs are fixed, new bugs are introduced
  — client code may need to be adapted

> Programmer solution: be egoistic

> “Get my bugfix in NOW, but change nothing else!”
Towards a Solution

> Software Engineering

> Process

> Language Design
Software Engineering

> Deprecation
> Modularizing
> Registration Mechanisms / Abstractions
> Refactoring
> Tests
Deprecation mechanism

> Retain old methods for compatibility
> But flag them as deprecated
>   — Raise warning at runtime

```smalltalk
Month>>eachWeekDo: aBlock
    self deprecated: 'Use #weeksDo:'.
    self weeksDo: aBlock
```

> retained for one release
> Problems:
>   — for methods only, change happens too often
Modularization

> 3.9: composed of 49 packages
  — average of 40 classes per package

> Has been done ad-hoc: need to be analysed!

> positive effects:
  — Packages maintained by third parties
  — Lots of hints where to clean up
New Abstractions

> Registration vs. editing code
  - Tools (e.g., refactoring), menus

> ToolBuilder: abstract the UI Framework

> System change notification
Refactoring and Tests

> refactoring
  — Remove prototype code
  — started to untangle packages

> tests
  — Programmer tests enable change
  — started to collect tests in 3.7
  — ~2000 in 3.9a
Process

- Better versioning tools
- Bug tracking
- Future: automatic build tools
Versioning Tools

> Old Smalltalk model: send patch files around
  — This does not scale!

> Monticello: versioning system for Squeak
  — contributed by the commercial sub-community
  — introduces simple package mechanism
  — very powerful merge tool
  — improves workflow
Bug Tracking

> No real bug tracking for a long time
  — Amazing! (but true for many projects)

> Introduced slowly around 3.7

> real tool based bug tracking since 3.8
Tests need to be executed to be useful
   — Squeak ships with many broken tests!

Solution: automatic test server
   — We are working on that now

Second step: automated build server
   — Build external packages
   — Run tests
Language Design

- Better support for Modularity
- History as a First Class Entity
- Beyond Deprecation
First class History

> Squeak is reflective: has a first class model of its own static structure (classes, methods)
> Extend the meta model to include data important for evolution
> History as a first class entity
  — Why did this change?
  — What else was change when this method changed?
  — When did this test break for the first time?
  — Which change affected the performance of the system?
> Deprecation: allow clients to migrate incrementally

> Can we do better?

> Complete history available
  — We can run different version per client
  — Slowly propagate changes through the system
Conclusion

> Evolution is a real problem for Squeak

> We need to improve on all levels
  — Better code base
  — Better tools + processes

> How can the language support evolution?
Questions?

> Evolution is a real problem for Squeak

> We need to improve on all levels
  — Better code base
  — Better tools + processes

> How can the language support evolution?