Techniques for Seamless Integration: A Talk from the Trenches

Julian Jones
IBM Canada Limited
Welcome to my world

Rational Software Architect
Rational Performance Tester
Rational Portfolio Manager
Rational Software Modeler
Rational Application Developer
Rational Functional Tester
Rational Web Developer
Rational Manual Tester

Analyst
Architect
Developer
Tester
Project Manager
Never has an advertisement struck so true

"Cat Herders" by Fallon advertising agency – spot created for EDS

161 Wizards in 28 categories

108 unique Features
Delivered in ~1400 Plugins

256 pages of Preferences in 34 categories

124 distinct Views

80 distinct Action Sets
The integration challenge

“lowest common denominator”

“each component for themselves”

Banal

Freakish
There are no silver bullets

But targeted use of the following techniques may take you some way towards ensuring that user interface contributions developed by semi-autonomous teams can be brought together as an integrated whole.
1. Stay current

- Be very familiar with the OS platform UI guidelines
  - revisit every year or so
- Be very familiar with the Eclipse UI guidelines
  - even though they are out of date
- Download the milestone builds
  - as well as the integration builds in the last 1/3rd of the release
  - play with the builds and study what’s new
- Watch the JDT project very closely
  - this project, along with the platform, typically establishes sound UI precedents to build upon
- Adopt and build on any cool innovations that you see
2. Don’t harden UIs prematurely

- Even Platform widgets should be considered on their merits
- For instance, it would have been very easy have anointed the v1 Properties view as the “chosen one” … but honestly speaking, it was too flawed to build anything useful around

DEVIVIATING FROM THE PLATFORM SHOULD BE DONE JUDICIOUSLY; but, by monitoring the pickup of various widgets by the platform and JDT teams, it becomes clear what UIs are playing out well (and what are getting left behind)
3. Facilitate visibility to emerging UI ideas

- Great UIs will propagate naturally … the “not invented here” syndrome quickly disappears when a strong UI is seen.
- For instance, the take up of PDE’s “flat-look” spread like wild-fire once word got out about it, and it was a no-brainer to get teams building editors that essentially parameterized highly structured documents to adopt this style.
4. Encapsulate proven UIs in frameworks

- As previously noted, great UIs speak for themselves and naturally spread within teams at the bleeding edge of development; but accelerated pickup and broad adoption truly comes about once the UI is encapsulated within a framework.

- Here is an example of the Property view that we ended up encapsulating within our UI frameworks.
5. Don’t be afraid of “natural selection”

- There will be times when there simply isn’t a clear winner between competing UIs. In such cases, emotions on all sides tend to run high. Rather than force the issue, one exceptional course of action is to let the market decide.

This is an easier path to take when the UIs have different targeted users. When the UIs have the same target users, internal indecision should – preferably – be kept internal.
6. But don’t discard centralized planning entirely

- Eclipse enables much to happen in a highly distributed fashion. But there are some things that fundamentally require highly centralized planning. Exploiting capabilities is one such example. The help system’s Table of Contents is another.
7. Let Eclipse step up to some challenges

- As a team building on top of Eclipse, there’s only so much that one can do; and some problems really ought to be solved at the platform level. Recognize these, and let Eclipse step up to those challenges.

- For instance, bringing some level of integration to the vast array of preferences that are possible is a huge drain on resources if done manually. By waiting until v3.1, there is the potential for a much cheaper and far more powerful solution.
In spite of everything Eclipse brings to the table, it is still not a perfect world and some areas remain broken from an integration point of view. The most obvious are in the area of menus (context menus and pull-down menus) where menu ordering is hard to control.

In such cases, there’s some hard-graft and manual work that can be done to minimize damages. But there should also be an attempt to push Eclipse to step up to even more. Fighting against menu ordering, for instance, is not a good use of one’s time.
## Summary

<table>
<thead>
<tr>
<th>Technique</th>
<th>Closing comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Stay current</td>
<td>Seems obvious; but few are. Also, unnecessary noise in design (and unnecessary invention) is reduced by having a common base of understanding</td>
</tr>
<tr>
<td>2. Don’t harden UIs prematurely</td>
<td>Consistency around a weak design isn’t something to be lauded; better to figure out what works first</td>
</tr>
<tr>
<td>3. Facilitate visibility to emerging UI ideas</td>
<td>It’s all about communication; once communication channels are set up, strong designs quickly propagate through them – people want to reuse strong design</td>
</tr>
<tr>
<td>4. Encapsulate proven UIs in frameworks</td>
<td>Can’t overestimate the power of frameworks</td>
</tr>
<tr>
<td>5. Don’t be afraid of “natural selection”</td>
<td>... though internal indecision should be kept internal</td>
</tr>
<tr>
<td>6. But don’t discard centralized planning</td>
<td>Also useful for product-wide “consistency” reviews (but see early comment about hardening UIs prematurely)</td>
</tr>
<tr>
<td>entirely</td>
<td></td>
</tr>
<tr>
<td>7. Let Eclipse step up to some challenges</td>
<td>Carefully consider the ROI; some integration issues simply aren’t worth pursuing through manual means</td>
</tr>
<tr>
<td>8. Push Eclipse to step up to even more!</td>
<td>Get active: leverage the newsgroups and mailing lists</td>
</tr>
</tbody>
</table>