Porting Suite of Swing Apps to e4 RCP

Robert Kellogg
Landmark Software and Services
Disclaimer

The content of this presentation is intended for informational purposes only and, unless otherwise agreed in writing, shall not be incorporated into any contract. Landmark makes no representation about the accuracy, reliability, completeness or timeliness of the information provided herein and the information may be revised at any time without notice. The development, release and timing of any features or functionality described for Landmark’s products remains at the sole discretion of Landmark. This presentation and the information herein is presented “as is” and Landmark makes no warranty, express or implied and Landmark assumes no liability in connection with the use of information contained herein.

Nothing in this presentation should be construed as granting any license or right to use the property of Landmark without the express written consent of Landmark and information presented herein may be subject to applicable existing non-disclosure agreements. The trademarks, service marks and logos displayed herein are the property of Landmark. All text, images, graphics and other materials in this presentation are subject to the copyright and other intellectual property rights of Landmark.
DecisionSpace® Software – Where Were We and Why Port?

- Evolved from separate apps sharing API to platform running plugin based apps together.
- Home grown module & UI framework based on swing.
- Need to expand platform SDK.
- Trying to leverage horizontal tools in our workflow.
- From 4 to 20 apps running together in single JVM.
- 5M+ lines of Java Code (45K Classes).
- 3M lines of native C++.
Eclipse 4 or 3?

- Port started in Oct 2011, e4.1.1 was out but it was still bumpy in places.
- Workbench paradigm fit our application model
- Pure e4 and emf could be a big learning curve for non computer science developers
- Can we leverage the workbench capabilities and e4 goodies
Eclipse 4

- CSS styling & styling tools
- e4 application hooks
- Change in view / editor policies
- IEEclipseContext – service collection
- Application Model
  - Editor rearranging/tiling – (no public APIs in 3.x)
  - Swing menu conversion to automatically update app model – didn’t work because of flickering and redraw issues.
- Dependency Injection
Modify Application Model to Tile Editors

```javascript
EModelService ems = window.getContext().get(EModelService.class);
MPerspective perspective = ems.getActivePerspective(window);
MUIElement elmt = ems.find(IPageLayout.ID_EDITOR_AREA, perspective);
MArea editorArea = (MArea) (elmt instanceof Mplaceholder) ? ((MPlaceholder) elmt).getRef() : elmt);

// set editor area’s sash container to horizontal or vertical
((MPartSashContainer)
  editorArea.getChildren().get(0)).setHorizontal(horizontal);

// loop through each MStackElement in the editor areas MPartStack

// select or create a new MPartStack to put retiled editors in

ems.insert(newStack, firstStack, horizontal ?
  EModelService.RIGHT_OF : EModelService.BELOW, 1.0f /
  (elements.size() + 1 - i));
```

Get the model, perspective, editor area from the active workbench window

Using EModelService insert each newStack in relation to the firstStack either horizontal or vertical. Each new stack's size is relative to the firstStack's and won't change again. So if we have N editors, then the first stack should have size 1/N, the second stack should have size 1/(N-1). Remember that elements.size() is off by one since the first element was removed.
SWT and Swing Integration

- All chrome, menus, toolbars converted to SWT
- Interior panels wrapped Swing
- SWT/AWT Bridge
  - Utility classes & methods
  - `runOnSWT()`, `invokeLater()`, `invokeAndWait()`
  - `AwtActionWrapper`, `AwtEventProcessor`
  - Modal dialogs and focus control – Albireo Project (see EclipseCon 2008 presentation)
Porting Challenges

- Dev culture and mentality
  - Their app isn’t the one and only
  - Other components may or may not be running
- Developers needed more hard docs than were available
  - Currently need to go under the hood of examples. Having to step through Eclipse code.
- Release and configuration mgmt challenges
- Automated UI testing having to be ported
- Long lead time for framework port
- Swing integration - still lots of wrapped UI code
- Workspace scope is different from working session, and having to work around
Keys to Success

- **OSGi 1st**
  - tooling & process

- **Extension Points**
  - Declarative extensions vs. programmatic allowing quicker application startup

- Swing Actions are flexible and allow misuse - converting to command/handlers forced cleanup of concerns

- Champions in the application teams

- Expert Consultants

- Unified UI paradigm
More Challenges to Come

- Long term life with Swing and SWT
- Graphics (still swing) moving to SWT event thread
  - Currently all apps interact on swing thread
  - Notification both in Swing & SWT for listeners. Reverse Bridge.
- Move 2D graphics to gef4??
Giving Back to the Community

- Better mgmt of embedded binaries.
- CSS tooling
- Bug fixes - platform UI bugs in the compatibility layer
- Eclipse context
- p2 stale dependency issues
Leveraging Other Eclipse Based Products and Projects

- Nebula - TableViewer
- EMF - model generation
- JBoss™ / Teiid
- Kettle™ - Spoon
- PENTAOHO®
- Activiti Designer™
Discussion & Questions?

Robert Kellogg at rkellogg@lgc.com

Thanks to:

- Brian deAlwis, Manumitting Technologies
- Joseph Peng, Halliburton – Landmark Software and Services