Re-structuring of a swing-based application into an Eclipse RCP

Christian Kurzke
Gustavo de Paula
Hugo Raniere
Agenda

• MOTODEV for JavaME Tools
• MOTODEV for JavaME SDK – Swing Version
• Swing Version – UI Workflow
• Swing Version – Class Diagram
• Issues to consider in porting to RCP
• Map UI Concepts
• Map Classes into Eclipse Plug-ins
• Some Metrics
• Conclusions
MOTODEV

• MOTODEV is the Motorola devices developer network

• Provides tool, documentation and support to developers

• Targets all different Motorola devices, from mobile devices to set-top boxes

• One of the main platforms supported inside MOTODEV is JavaME
MOTODEV for JavaME SDK – Swing Version

- SDK that can be integrated on any UEI compliant IDE
- Swing Application
- Includes an UEI JavaME Emulator
- Support most of MIDP 2.0 Motorola devices
- Includes complete documentation about the Devices and the APIs
- Include external tools that support the development
- Includes demo of JavaME APIs
Swing Version – UI Workflow

Tab selection

• Each tab is mapped to a swing panel
• Each tab presents a different set of functions
  - Home: main tab
  - Launch: launch emulator
  - Docs: help
  - Services: emulator services
  - Tools: external tools
  - About: about dialog
Issues to consider in porting to RCP

• Map of UI Concepts
• Map of Functionalities
• Map of Classes into Plug-ins
Map UI Concepts

- Tried to keep as most as possible the original user experience
  - Reduce the impact to the end-user

- All UI concepts available on Swing are also available on SWT/JFaces

- Some concepts were adapted in order to keep the same user experience
  - “Swing Tabs”
  - External Dialogs
Map UI Concepts – “Tabs”

Tabs were mapped to Eclipse perspectives

Re-structuring of a swing-based application into an Eclipse RCP | © 2008 by Motorola Inc.; made available under the EPL v1.0
Map UI Concepts – “Service Tab”

Services were open in Run button
Map UI Concepts – “External Dialogs”

External dialogs were turned into Eclipse views
Map of Functionalities

• Most of the functionalities are the same, but Eclipse RCP brings some services that can be reused

• “About tab” was removed and “Eclipse About” was used

• “Docs tab” was removed and “Eclipse Help” was used
Map Classes into Eclipse Plug-ins

• Original Swing application already separated UI from business logic

• Most of the code can be re-used

• If UI and business logic are not separated, most of the code might have to be refactored

• Three plug-ins were defined
  • Core: encapsulate all business logic classes
  • RCP: encapsulate Eclipse application itself
  • UI: encapsulate the UI
Swing Version – Class Diagram

Swing dependent code

presentation

util
data

business
Some Metrics…

- Number of Classes is similar in both versions
  - +10% increase in RCP version

- Code size is similar in both versions
  - +10% increase in RCP version

- Final deployment size increased in RCP
  - Eclipse RCP has to included
  - But services like update manager and Eclipse Help System are now available on the tool

- Effort to re-structure application into a RCP was +4 staff/month
Conclusions

• Re-structure of a Swing application into a RCP depends highly on how the original application is organized

• Effort can be low if the UI is separated

• All UI concepts available on Swing can be implemented on SWT/JFaces

• Some functionality can be replaced by already existing Eclipse Services