Experiences with Eclipse Rich Client Application Development

Showcase: GDF SUITE Management Center

Feb. 4\textsuperscript{th} 2004
Dr. Frank Gerhardt, Chris Wege
Overview

• What is the GDF SUITE Management Center?
• What does it do?
• Demo
• How does it work?
• How did we do it?
• Experiences
• Rich Client Platform
• Wish List
GDF SUITE

Overview

GDF SUITE

Features

Demo

Implementation

Development

Experiences

RCP and Future

Wish List

References

3

Experiences

with Eclipse

Rich Client

Application

Development

© Frank Gerhardt

and Chris Wege

• **GDF** = Geographic Data Files
  • ISO standard
  • File format for exchanging geo data

• **GDF SUITE** by **LOGIBALL**
  • Business solution for processing GDF data
  • Based on ORACLE with Spatial Extension
  • Long-running processes (days-weeks) on large geo databases (tens of GBs)

• **GDF SUITE Management Center**
  • Eclipse-based rich client application for management of geo databases and processes
GDF SUITE Management Center

- Directory of GDF databases (e.g. many DB instances for different countries)
  - Hierarchical organization into folders (e.g. Europe/Germany/Bavaria/Oktoberfest)

- Process Management
  - Starting, stopping
  - Progress monitoring
  - Logging, error handling

- Viewing GDF data
  - Map (zoom, pan, select)
  - GDF Features (e.g. one-way road info)
Demo (RDCP Perspective)

Overview
GDF SUITE
Features
Demo
Implementation
Development
Experiences
RCP and Future
Wish List
References

Experiences with Eclipse Rich Client Application Development

© Frank Gerhardt and Chris Wege

© LOGIBALL GmbH
Demo (SUD Perspective)
Why Eclipse?

- Multi-Platform (Solaris, Linux, Win32)
- Open Source, CPL permits redistribution
- Look and feel, speed of GUI (SWT)
- Plugin architecture
  - Separation of base product from customer-specific features
  - Base for licensing by feature
- Good software design of Eclipse
- Branding of primary feature
GDF SUITE Management Center
Platform Architecture (1)

- Based on Eclipse SDK 2.1.1 (keeping an eye on 3.0 milestone builds)
  - Release, not a milestone build, management buy-in
  - GDF SUITE MC targeted for February 2004
  - No migration to 2.1.2 because of no added value
- Several plug-ins, separate test plug-in
- Two perspectives
- Eight views, six wizards
- No editors, no resources
Eclipse Features Used

- Eclipse SDK (not platform), many plugins disabled (see next slide)
- GEF, Draw2D
- Runtime, SWT, JFace, Workbench
- Preferences, Encryption, Wizards, Progress, Icons, Branding
Eclipse Features NOT Used

- Workspace (apart from preferences)
- Team
- JDT (but compiler still enabled for help)
- Removed several menu entries related to resources and Java development

→ Needed to disable some features in the Eclipse source code
Example: Removing a Menu Item

- Needed to remove hard-coded (2.1.1) functionality in our Eclipse build

```java
if (this.enabled) {
    // Add new project ...
    innerMgr.add(newProjectAction);
}
```

Removed New Project from Navigator context menu
Development process

- 4 Developers, 2 locations, 6 months
- 3-week iterations with management reviews
- Test-Driven Development, XP'ish, Pair-Programming
- PDE JUnit tests in separate plug-in
- Bugzilla
Build process

- CVS contains Eclipse SDK 2.1.1, GEF/Draw2d, and GDF SUITE MC
- Incremental build
  - Using PDE
  - Short-running tests with PDE JUnit
- Nightly build
  - Derived from Eclipse source dist
    - Builds custom Eclipse version
    - Reused build setup for building our primary feature
  - Runs also long-running tests
  - Reports
• Testing the GUI programmatically is easy with PDE JUnit, e.g. New Layer Wizard

```java
public void testPerformFinish() throws Exception {
    String viewId = IConstants.ID_LAYERVIEW;
    double zoomFactor = 1;

    IWorkbenchPage activePage = DatabaseUiPlugin.getDefault().getWorkbench().getActiveWorkbenchWindow().getActivePage();
    SUDView sudView = (SUDView) activePage.showView(viewId);
    LayerManager layerManager = sudView.getLayerManager();
    int oldSettingsCount = layerManager.getLayerSettings(zoomFactor).length;

    NewLayerWizard wizard = new NewLayerWizard(view);
    TestableWizardDialog dialog = openWizardDialog(wizard);
    NewLayerWizardPage page = (NewLayerWizardPage) wizard.getPages()[0];
    page.setLayerName("Test Layer");
    page.setFeatureCategory("1");
    page.setFeatureClass("1111");
    page.setAttributeType("someType");
    page.setAttributeValue("someValue");
    dialog.finishPressed();
    LayerSetting[] newLayerSettings = layerManager.getLayerSettings(zoomFactor);
    int newSettingsCount = newLayerSettings.length;
    assertEquals(oldSettingsCount + 1, newSettingsCount);
    LayerSetting retrievedSetting = newLayerSettings[newSettingsCount - 1];
    assertEquals("Test Layer", retrievedSetting.getLayerName());
}
```
Experiences (1)

+ Fun, productive
+ Could reuse a lot

• Design
• Development process (nightly builds and reports)
• Many small goodies (icons, splash screen, etc.)
Experiences (2)

+ Views and perspectives fit our application well
+ No complex layout manager used
+ No GUI builder needed
+ All configured in XML
+ Testing with PDE Unit useful
+ Monkey see, monkey do: Reading Eclipse source code helped a lot to learn and understand
+ Book “Contributing to Eclipse” very helpful, draft was available very early
Experiences (3)

- Needed to change some Eclipse code
- Needed our own build of Eclipse code
- Documentation is fragmented
- Hard to kick out compiler
- Draw2D Koordinaten-Modell insufficient (e.g. translateToParent())

Desired improvements on our side:
- No build of Eclipse
- No distribution of compiler
Recommendations (1)

• Do
  – “Use the Source, Luke”
  – Read the books
  – Read and post to newsgroup
  – Read and post bug reports
  – Start with one plug-in and refactor
  – Keep your design compatible with future Eclipse releases:
    • Investigate milestone builds
    • Know Eclipse roadmap
Recommendations (2)

- Do NOT
  - Start with a complex plug-in structure
    - Evolve plug-in structure
  - Duplicate platform functionality
    - Read, read, read
With Eclipse 3.0 and RCP

- No Eclipse source code changes
  - No own Eclipse build any more
  - Nightly build only for our own code
- Many necessary features were already available in Eclipse 2.1.1 but buried in IDE or in internals
  - However: We will stick to our own process manager
- RCP might have taken out too much (e.g. extension points for menus)
Future Plan

- Update site
- Multi-user, authentication, authorization
- Extension points for database types and process types
  - More plug-ins at these points
  - Establishing GDF SUITE as a platform for geodata applications (inhouse and external)
- GDF SUITE Development Kit
- Migration to 3.0 after 3.0 release (Q3)
Wish List

- One distribution for all platforms
- Cross-platform installer / uninstaller
- WebStart, smaller size
- Java Help, or PDF authoring solution for Eclipse Help
- Background process management
- License management (e.g. support trial versions)
- Authentication, authorization, user management, user profiles, integration with update manager
References

- LOGIBALL GDF SUITE
  www.logiball.de
- Eclipse source code
- Shavour et al.: *The Java Developer's Guide to Eclipse*
- Gamma, Beck: *Contributing to Eclipse*
Thank You for Your Attention

- Dr. Frank Gerhardt
  fg@SoftwareExperts.de
  www.FrankGerhardt.com

- Chris Wege
  wege@acm.org
  purl.org/net/wege