**Projekthandbuch**

Eclipse RCP ist ein Standardframework für Geschäftsanwendungen. Mit der neusten Generation E4 wurde Eclipse RCP vollständig modernisiert. Anhand einer wichtigen RCP Applikation der SBB wird eine Migration auf Eclipse E4 exemplarisch durchgeführt und die dabei berücksichtigten Aspekte dargestellt.

|  |  |
| --- | --- |
| Autor: | Mike Rothenbühler |
| Version: | 0.1 |
| Status: | In Arbeit |
| Ablage: | https://github.com/MikeR13/MAS/blob/master/Deliverables/ |
| Institution: | Hochschule für Technik und Informatik Bern |
| Verteiler: | Brawand Ueli, Hoffmann Marc, Rothenbühler Mike |

**Versionkontrolle**

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Version** | **Autor** | **Bemerkungen** |
| 03.06.2013 | 0.1 | MIRO | Erster Wurf |
|  |  |  |  |

Inhaltsverzeichnis

1. Einleitung 6

1.1. Zweck des Dokumentes 6

1.2. Problemstellung 6

1.3. Randbedingungen 6

1.4. Situationsanalyse 6

1.5. Erbrachte Vorleistung 6

2. Vorbereitung 7

2.1. Architektur Eclipse 3 vs 4? Oder eher im Aspekt Nr.1 7

3. Organisatorisches 7

4. Projektstart 7

4.1. Ziele 7

5. Aspekt Iterationen 7

5.1. Beschreibung des Aspektes „Mixing E3 / E4“ 7

5.2. Diskussion der Eclipse RCP 4 Lösung 7

5.2.1. Vorteile 7

5.2.2. Vergleich mit Eclipse RCP 3 7

1. How does Eclipse 4.x differ from 3.x architecturally? (from http://www.eclipse.org/eclipse/development/porting/4.2/faq.php ) 7

5.2.3. Einschränkungen und Risiken 7

5.2.4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3 8

5.3. Konkretes Beispiel RCS 8

5.3.1. Definition Abnahmekriterien 8

5.3.2. Migration 9

5.3.3. Test 9

5.4. Beschreibung des Aspektes „Adapters / Dependency Injection“ 9

5.5. Diskussion der Eclipse RCP 4 Lösung 9

5.5.1. Vorteile 9

5.5.2. Vergleich mit Eclipse RCP 3 9

5.5.3. Einschränkungen und Risiken 9

5.5.4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3 9

5.6. Konkretes Beispiel RCS 9

5.6.1. Definition Abnahmekriterien 9

5.6.2. Migration 9

5.6.3. Test 9

5.7. Beschreibung des Aspektes „Commands / Handler, Menus, Key Bindings“ 9

5.8. Diskussion der Eclipse RCP 4 Lösung 9

5.8.1. Vorteile 9

5.8.2. Vergleich mit Eclipse RCP 3 9

5.8.3. Einschränkungen und Risiken 9

5.8.4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3 9

5.9. Konkretes Beispiel RCS 9

5.9.1. Definition Abnahmekriterien 9

5.9.2. Migration 9

5.9.3. Test 9

5.10. Beschreibung des Aspektes „Eigene Extension Points / Eigene Services“ 9

5.11. Diskussion der Eclipse RCP 4 Lösung 9

5.11.1. Vorteile 9

5.11.2. Vergleich mit Eclipse RCP 3 9

5.11.3. Einschränkungen und Risiken 9

5.11.4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3 10

5.12. Konkretes Beispiel RCS 10

5.12.1. Definition Abnahmekriterien 10

5.12.2. Migration 10

5.12.3. Test 10

5.13. Beschreibung des Aspektes „Application Model vs. Advisors“ 10

5.14. Diskussion der Eclipse RCP 4 Lösung 10

5.14.1. Vorteile 10

5.14.2. Vergleich mit Eclipse RCP 3 10

5.14.3. Einschränkungen und Risiken 10

5.14.4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3 10

5.15. Konkretes Beispiel RCS 10

5.15.1. Definition Abnahmekriterien 10

5.15.2. Migration 10

5.15.3. Test 10

5.16. 10

6. Reflexion 10

Einleitung

* 1. Zweck des Dokumentes

Dieses Projekthandbuch informiert die an „Migration von Eclipse 3.x nach Eclipse 4“ beteiligten Parteien über die getätigten Migrationen von den ausgewählten Aspekten.

* 1. Problemstellung

Da sich mit der Version 4 einiges an Eclipse RCP geändert hat ist eine Migration nicht einfach so zu bewerkstelligen. Es gibt aus der Community (noch) nicht viele Berichte zu gelungenen Migrationen, geschweige denn eine Anleitung wie eine solche Migration erfolgreich durchgeführt werden kann.

Es sollen Erkenntnisse gewonnen werden, wie eine erfolgreiche Migration durchgeführt werden kann, ohne dass die bestehende Applikation in den Punkten

* Funktionalität
* Performance
* Stabilität
* Usability
* Look and Feel

negativ beeinflusst wird. Die Arbeit an der bestehenden Applikation soll auch während der Migrationszeit möglich sein. Hierfür müssen Lösungen erarbeitet werden.

Mit den Erfahrungen und Ergebnissen aus der Master Thesis soll eine Migration auch für grosse Projekte relativ einfach möglich sein.

* 1. Randbedingungen

Das Projekt wird im Rahmen der Semesterarbeit und der Diplomarbeit durchgeführt. Dafür gelten die an der HTI üblichen Bedingungen.

* 1. Situationsanalyse

Um die erarbeiteten Migrationsvorschläge in der Praxis zu verifizieren, wird die Migration des RCP-Clients exemplarisch durchgeführt.

Bei dem RCS Client handelt es sich um die grösste Eclipse RCP Applikation der SBB mit folgenden Kennzahlen:

* 70 Plugins
* über 10 Hauptfenster
* Dutzende von Dialogen
* 270‘000 Zeilen Code

RCS wird zur Disposition des Zugverkehrs auf dem gesamten Streckennetz der SBB verwendet.

* 1. Erbrachte Vorleistung

Das Knowhow über Eclipse RCP 4 wurde bereits vor Projektstart aufgebaut.

Vorbereitung

* 1. Architektur Eclipse 3 vs 4? Oder eher im Aspekt Nr.1

Organisatorisches

Projektstart

* 1. Ziele

Aspekt Iterationen

* 1. Beschreibung des Aspektes „Mixing E3 / E4“

In dieser Iteration soll geprüft werden welche Migrationsmöglichkeiten überhaupt existieren. Können Eclipse RCP 3 und Eclipse RCP 4 Komponenten im selben Projekt gleichzeitig nebeneinander im Einsatz sein?

* 1. Diskussion der Eclipse RCP 4 Lösung
     1. Vorteile

Architektur?

* + 1. Vergleich mit Eclipse RCP 3

Architektur?

**How does Eclipse 4.x differ from 3.x architecturally? (from** <http://www.eclipse.org/eclipse/development/porting/4.2/faq.php> )

The Eclipse SDK 4.2, for the most part, contains all the plug-ins that make up Eclipse 3.7. That is, all of Java development tools (JDT) and Plug-in Development Environment (PDE), and most of the Platform, are the same bits as in 3.7. What's different is the implementation of the Workbench (org.eclipse.ui.workbench plugin), and the technologies this new implementation is based on. Before the release, the technologies (modeled user interface, dependency injection and service-based programming model, CSS-based styling) were called 'e4' but we are now referring to them as the Eclipse 4 Application Platform. On top of the Eclipse 4 Application Platform, the 4.x Workbench offers an implementation of the 3.x Workbench APIs, to provide backwards compatibility for the Eclipse IDE and its plug-ins.

* + 1. Einschränkungen und Risiken

Verweis auf Risiken im Projektbericht?

* + 1. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3

Pojos vs. ViewParts etc..

Wohin kommmt das?:

**Option 1: Compatibility Layer einsetzen**

<http://eclipsesource.com/blogs/2012/06/18/migrating-from-eclipse-3-x-to-eclipse-4-e4/>

Der Compability Layer ermöglicht es Eclipse RCP 3.x Applikationen ohne Codeanpassungen auf der Eclipse 4 Plattform zu laufen. Wenn man nicht auf Eclipse 4 migriert so bleibt man kompatibel mit 3.x.

Um die Migration zu erleichtern bietet der Compability Layer the 3.x Workbench APIs an und übersetzt alle Aufrufe in das Programmiermodell von E4. Im Hintergrund wird transparent ein Application Model erstellt. Eclipse 3.x Applikationen sollten jedoch keine interne Workbench APIs benutzen um mit dem Compability Layer zu funktionieren.

Eclipse 4 benötigt aber zusätzlich die folgenden Plugins um zu funktionieren:

* org.eclipse.equinox.ds
* org.eclipse.equinox.event
* org.eclipse.equinox.util
* org.eclipse.e4.ui.workbench.addons.swt

Mit dem Compability Layer Ansatz alleine kann man aber nicht von den neuen Konzepten von E4 (DI und Annotationen) Gebrauch machen. CSS funktioniert mit diesem Ansatz.

**Option 2: Eine reine Eclipse RCP 4 Applikation**

Diese Option wird nicht betrachtet, da ja eine bestehende Eclipse RCP 3.x Applikation migriert und nicht neu geschrieben werden soll.

**Option 3 Compatibility Layer und Eclipse RCP 4 Plugins**

Hier werden neue Plugins in Eclipse RCP 4 Manier geschrieben, die alten werden auf Eclipse 3.x belassen und laufen koexistent auf dem Compability Layer.

Es gibt 3 Arten Eclipse RCP 4 Plugins im Compability Layer zu integrieren.

1. Um dem Application Model – welches vom Compability Layer erstellt wird- Elemente hinzuzufügen werden Prozessoren und Fragmente benutzt. Hier gibt es aber aktuell noch Timing Probleme, denn wenn die Prozessoren und Fragmente verarbeitet werden hat der Compability Layer das Application Model noch nicht komplett erstellt. Diese Option mag für Handles und View funktionieren aber nicht für Editors.
2. Das Application Model, das vom Compability Layer erstellt wird, wird kopiert und als Application Model registriert. Diesem Application Model können nun neue Eclipse RCP 4 Komponenten hinzugefügt werden. Das Model XMI-File – konkret: LegacyIDE.xml - kann aus dem Plugin org.eclipse.ui.workbench herauskopiert werden. **🡪 So wie’s aussieht ist das jetzt das workbench.xmi im Plugin org.eclipse4.workbench!   
   TODO:   
   Hier alle Schritte aufführen welche nötig sind um das Ganze (workbench.xmi) gebrauchen zu können.  
   Evtl. XMI splitten / Komponenten, wie???**
3. Die 3.x e4-Bridge von Tom Schindl The goal of the bridge is to ease single sourcing applications on 3.x and e4, which means that views and editors can be used in 3.x and e4 in parallel. To enable this, the plugin org.eclipse.e4.tools.compat provides wrapper classes that implement the interfaces of 3.x. For example, the wrapper DIViewPart implements ViewPart. In the wrapper, you specify a class (POJO), which implements a view following the e4 programming model, including dependency injection. Essentially the wrapper is just a pointer to an e4 object. It will initialize the POJO using dependency injection.
4. A 3.x wrapper
5. public class ExampleViewWrapper extends DIViewPart{
6. public Example3xViewWrapper() {
7. super(ExampleE4View.class);
8. }
9. }
11. A e4 view:
12. public class ExampleView {
13. private Label label;
14. @Inject
15. public ExampleView(Composite parent){
16. label = new Label(parent, SWT.NONE);
17. label.setText("Hello World");
18. }
19. }
21. This approach allows you to develop new parts of the application using all the benefits of e4 and as well, reuse all existing components. Further, the views developed in this way can be integrated into any pure e4 application without any adaptations. The approach is described more in detail in this tutorial. [this tutorial](http://eclipsesource.com/blogs/tutorials/eclipse-4-e4-tutorial-soft-migration-from-3-x-to-eclipse-4-e4/).
    1. Konkretes Beispiel RCS
       1. Definition Abnahmekriterien
       2. Migration
       3. Test
    2. Beschreibung des Aspektes „Adapters / Dependency Injection“
    3. Diskussion der Eclipse RCP 4 Lösung
       1. Vorteile
       2. Vergleich mit Eclipse RCP 3
       3. Einschränkungen und Risiken
       4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3
    4. Konkretes Beispiel RCS
       1. Definition Abnahmekriterien
       2. Migration
       3. Test
    5. Beschreibung des Aspektes „Commands / Handler, Menus, Key Bindings“
    6. Diskussion der Eclipse RCP 4 Lösung
       1. Vorteile
       2. Vergleich mit Eclipse RCP 3
       3. Einschränkungen und Risiken
       4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3
    7. Konkretes Beispiel RCS
       1. Definition Abnahmekriterien
       2. Migration
       3. Test
    8. Beschreibung des Aspektes „Eigene Extension Points / Eigene Services“
    9. Diskussion der Eclipse RCP 4 Lösung
       1. Vorteile
       2. Vergleich mit Eclipse RCP 3
       3. Einschränkungen und Risiken
       4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3
    10. Konkretes Beispiel RCS
        1. Definition Abnahmekriterien
        2. Migration
        3. Test
    11. Beschreibung des Aspektes „Application Model vs. Advisors“
    12. Diskussion der Eclipse RCP 4 Lösung
        1. Vorteile
        2. Vergleich mit Eclipse RCP 3
        3. Einschränkungen und Risiken
        4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3
    13. Konkretes Beispiel RCS
        1. Definition Abnahmekriterien
        2. Migration
        3. Test
22. Reflexion