**Projektbericht**

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** |
| 01.06.2013 | 0.1 | MIRO | Erster Wurf |
| 06.07.2013 | 0.2 | MIRO | Berichte für die ersten beiden Aspekte |

Inhaltsverzeichnis

1. Einleitung 6

1.1. Zweck des Dokumentes 6

2. Projekt / Projektziele 6

2.1. Umfeld 6

2.2. Stakeholder 6

2.3. Zielvorstellungen 6

2.4. Lösungen 6

2.5. Sicherheits- und Datenschutzaspekte 7

3. Risiken 8

3.1. Risikoidentifizierung, -bewertung und -quantifizierung 8

3.2. Risikobehandlung 9

4. Vorbereitung 10

5. Organisatorisches 11

6. Projektstart 11

6.1. Ziele 11

7. Aspekt Iterationen 11

7.1. Beschreibung des Aspektes 11

7.2. Diskussion der Eclipse RCP 4 Lösung 11

7.2.1. Vorteile 11

7.2.2. Vergleich mit Eclipse RCP 3 11

7.2.3. Einschränkungen und Risiken 11

7.2.4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3 11

7.3. Konkretes Beispiel RCS 11

7.3.1. Definition Abnahmekriterien 11

7.3.2. Migration 11

7.3.3. Test 11

7.4. Beschreibung des Aspektes 11

7.5. Diskussion der Eclipse RCP 4 Lösung 11

7.5.1. Vorteile 11

7.5.2. Vergleich mit Eclipse RCP 3 11

7.5.3. Einschränkungen und Risiken 11

7.5.4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3 11

7.6. Konkretes Beispiel RCS 11

7.6.1. Definition Abnahmekriterien 11

7.6.2. Migration 11

7.6.3. Test 11

7.7. Beschreibung des Aspektes 12

7.8. Diskussion der Eclipse RCP 4 Lösung 12

7.8.1. Vorteile 12

7.8.2. Vergleich mit Eclipse RCP 3 12

7.8.3. Einschränkungen und Risiken 12

7.8.4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3 12

7.9. Konkretes Beispiel RCS 12

7.9.1. Definition Abnahmekriterien 12

7.9.2. Migration 12

7.9.3. Test 12

7.10. Beschreibung des Aspektes 12

7.11. Diskussion der Eclipse RCP 4 Lösung 12

7.11.1. Vorteile 12

7.11.2. Vergleich mit Eclipse RCP 3 12

7.11.3. Einschränkungen und Risiken 12

7.11.4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3 12

7.12. Konkretes Beispiel RCS 12

7.12.1. Definition Abnahmekriterien 12

7.12.2. Migration 12

7.12.3. Test 12

7.13. Beschreibung des Aspektes 12

7.14. Diskussion der Eclipse RCP 4 Lösung 12

7.14.1. Vorteile 12

7.14.2. Vergleich mit Eclipse RCP 3 12

7.14.3. Einschränkungen und Risiken 12

7.14.4. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3 12

7.15. Konkretes Beispiel RCS 12

7.15.1. Definition Abnahmekriterien 13

7.15.2. Migration 13

7.15.3. Test 13

7.16. 13

8. Reflexion 13

Einleitung

* 1. Zweck des Dokumentes

Dieser Projektbericht informiert die an „Migration von Eclipse 3.x nach Eclipse 4“ beteiligten Parteien über den Verlauf und die Ergebnisse des Projektes.

Projekt / Projektziele

* 1. Anlass und Begründung des Projektes

Die SBB IT haben in mehreren Projekten Eclipse 3.x Rich Client Platform (im Folgenden RCP genannt) im Einsatz. Die neue Version Eclipse 4 RCP wurde bereits in mehreren neueren Projekten eingesetzt und hat sich bewährt.

Eclipse RCP kurz erklärt (aus <http://de.wikipedia.org/wiki/Eclipse_(IDE))>:

„*Eclipse bietet die Rich Client Platform, welche es Anwendungsentwicklern ermöglicht, basierend auf dem Eclipse Framework, von der Eclipse-IDE unabhängige Anwendungen zu schreiben. Eine Übersicht befindet sich auf der Eclipse-Homepage.[<http://eclipse.org/community/rcp.php>] Beispielsweise basiert IBM Workplace Clients auf Eclipse RCP.*

*Die folgenden Komponenten (Plug-ins) werden mindestens für ein Eclipse-Rich-Client-Platform-Programm benötigt:*

* *Eclipse Core Platform – steuert den Lebenszyklus einer Eclipse-Application*
* *Standard Widget Toolkit (SWT)*
* *JFace*

*Weitere Eclipse-Komponenten, wie das Hilfesystem oder das automatische Aktualisierungssystem, können auch eingesetzt werden. Üblicherweise wird das Eclipse-OSGi-Framework Equinox zum Kombinieren („bundling“) der Komponenten eingesetzt.*“

Das Rail Control System (RCS) ist eine der grössten Eclipse RCP Anwendungen bei den SBB. RCS soll mindestens noch 10 Jahre im Einsatz sein und kontinuierlich ausgebaut werden. Eclipse 3 wurde deprecated, es findet keine Weiterentwicklung mehr statt. RCS muss also auf Eclipse RCP 4 migriert werden.

Eclipse RCP 4 bietet ein moderneres Programmiermodell an als sein Vorgänger. Die vielfältigen APIs aus RCP 3.x wurden deutlich reduziert und vereinheitlicht. Die über das gesamte API verteilten Singletons wurden entfernt.

Neben der Vereinfachung wurden moderne Konzepte wie Dependency Injection und Declarative Styling eingeführt.

Die Implementation von Rich Client Applikation in Eclipse RCP wird mit der Version 4 flexibler und deutlich vereinfacht. Die Produktivität der Programmierer steigt, die Testbarkeit und die Wartung der Applikationen wird erleichtert.

* 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.

* 1. Stakeholder

|  |  |  |
| --- | --- | --- |
| **Rolle** | **Name** | **Email** |
| Projektumsetzung | Mike Rothenbühler | [michael.rothenbuehler@sbb.ch](mailto:michael.rothenbuehler@sbb.ch) |
| Projektbetreuer | Marc Hoffmann | [marc.hoffmann@sbb.ch](mailto:marc.hoffmann@sbb.ch) |
| Experte | Ueli Brawand | [ueli.brawand@besonet.ch](mailto:ueli.brawand@besonet.ch) |

* 1. Zielvorstellungen

Das Ziel des Projektes ist es mindestens 5 Aspekte von Eclipse RCP 3 auf Eclipse 4 RCP zu migrieren. Als Aspekt wird zum Beispiel das plugin.xml, die Selektion oder auch Dependency Injection betrachtet.

Für jeden Aspekt soll folgendes durchgeführt werden:

Konzept

* Beschreibung des Aspektes
* Diskussion der Eclipse RCP 4 Lösung
  + Vorteile
  + Vergleich mit Eclipse RCP 3
  + Einschränkungen und Risiken
  + Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3

Konkretes Beispiel RCS

* Definition Abnahmekriterien
* Migration
* Test

Die Erfahrungen im Projekt sollen in einem Bericht geschildert werden.

* 1. Lösungen

Die Lösung eines Aspektes wird jeweils während einer Iteration erarbeitet. Nach dem heutigen Wissenstand gibt es noch keine möglichen Lösungsvarianten. Nach Möglichkeit sollen jedoch dokumentierte „Best practices“ zum Einsatz kommen (zum Beispiel aus „*Eclipse 4 - Rich Clients mit dem Eclipse 4.2 SDK*“ von Marc Teufel und Jonas Helming oder „*Eclipse 4 Application Development*“ von Lars Vogel).

* 1. Sicherheits- und Datenschutzaspekte

Der Code der SBB darf nicht an Dritte gelangen. Um dies zu verhindern wird der Code ausschliesslich im SBB Netz bearbeitet und dann höchstens Ausschnitte daraus publiziert.

Risiken

* 1. Risikoidentifizierung, -bewertung und -quantifizierung

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Id** | **Risiko** | **Eintritts-wahrscheinlichkeit** 0 - 15 % unwahrscheinlich 15 - 30 % eher unwahrscheinlich 31 - 50 % möglich 51 – 100 % wahrscheinlich | **Begründung** | **Auswirkung des Risikos aufs Projekt** 8 kritisch  4 gross 2 klein 1 vernachlässigbar | **Begründung** |
| 1 | Migration wegen unbekannten technischen Problemen nicht möglich | 20 % | Mit dem heutigen Wissenstand darf davon ausgegangen werden, dass es keine technische Probleme geben sollte. | 8 | Das Projekt könnte nicht durchgeführt werden, die Migration wäre gescheitert |
| 2 | Zeitplan kann nicht eingehalten werden | 40 % | Der Zeitplan ist sehr sportlich ausgelegt, es steht eine sehr kurze Zeit für die Bearbeitung des Projektes zur Verfügung. | 4 | Einzelne Aspekte könnten nicht migriert werden. |
| 3 | Funktionalität geht verloren | 40 % | Eclipse RCP 4 bietet heute noch nicht alle Funktionen von Eclipse RCP 3 an. | 8 | Es darf unter keinen Umständen Funktionalität verloren gehen, das würde vom Auftraggeber nicht akzeptiert werden. |
| 4 | Performanceeinbussen | 50 % | Es gibt Berichte darüber, dass Eclipse 4 teilweise mit Perfomanceproblemen zu kämpfen hat. | 8 | Die Applikation muss mindestens genauso performant sein wie heute. Etwas anderes wir vom Auftraggeber nicht akzeptiert |
| 5 | Verlust von Stabilität | 20 % | Es sind heute keine Probleme betreffen Stabilität von Eclipse RCP 4 bekannt. | 8 | Eine abstürzende Applikation oder dergleichen ist absolut inakzeptabel |
| 6 | Verlust von Usability | 10 % | Geringe Wahrscheinlichkeit da die Eclipse IDE seit Version 3.8 standardmässig auf E4 ausgeliefert wird. | 8 | RCS ist das wichtigste Arbeitswerkzeug der Disponenten und muss somit eine hohe Usability aufweisen. |
| 7 | Wartbarkeit nimmt ab | 10 % | In der Wartbarkeit von Eclipse RCP 4 Applikationen erwarten wir eine eklatante Verbesserung gegenüber Eclipse RCP 3. | 4 | Das Projekt darf in der Wartung keinesfalls teurer werden, da die Wartungskosten bereits heute zu hoch sind. |
| 8 | Testbarkeit nimmt ab | 10 % | Auch betreffend Testbarkeit dürfte Eclipse RCP 4 um einiges besser dastehen als Eclipse RCP 3. | 4 | Um die Qualität der Applikation zu gewährleisten muss diese auch testbar sein bzw. bleiben. |
| 9 | Look and Feel wird von Anwendern nicht toleriert | 10 % | Das Look and Feel wird wohl ziemlich gleich bleiben. | 4 | Das Look and Feel sollte gleich bleiben da Änderungen gegebenenfalls zu Schulungen und Anpassungen von Handbüchern führen. |

* 1. Risikobehandlung

|  |  |  |
| --- | --- | --- |
| **Id** | **Risiko** | **Risikobehandlung** |
| 1 | Migration wegen unbekannten technischen Problemen nicht möglich | Es ist umfangreiches Eclipse Knowhow im Projekt vorhanden und es bestehen Kontakte zur Eclipse-Community. |
| 2 | Zeitplan kann nicht eingehalten werden | Die Behandlung eines Aspektes soll timeboxed erfolgen. Das heisst, dass pro Aspekt eine maximale Anzahl an Stunden zur Verfügung steht und diese Zeit darf nicht überschritten werden. Wenn ein Aspekt innerhalb dieses Zeitraumes nicht erfolgreich migriert werden konnte, dann ist das nun einmal so. Die Erfahrungen werden dokumentiert und es soll kritisch hinterfragt werden, ob man die Migration des Aspektes unter Umständen andersartig hätte angehen müssen. |
| 3 | Funktionalität geht verloren | Um dieses Risiko zu minimieren, soll jeweils vor und auch nach der Migration die Funktionalität überprüft und dokumentiert werden. Die beiden Überprüfungen sollten zum selben Resultat führen. . Bestehende funktionale Testfälle müssen durchgeführt werden. |
| 4 | Performanceeinbussen | Im RCS Client sind diverse Performanceprüfungen eingebaut. Bestimmte ausgewählte Kennzahlen sollen vor und nach der Migration erfasst und miteinander verglichen werden. Falls Differenzen bestehen werden diese kritisch hinterfragt und, wenn nötig, Massnahmen eingeleitet. Bestehende nichtfunktionale Testfälle müssen durchgeführt werden. |
| 5 | Verlust von Stabilität | Nach der Migration eines Aspektes soll jeweils durch gezielte manuelle Tests festgestellt werden, ob die Applikation nach wie vor stabil läuft. |
| 6 | Verlust von Usability | Bei Änderungen der Bedienung müssen diese gemeinsam mit den Anwendern verifiziert werden. |
| 7 | Wartbarkeit nimmt ab | Die Wartbarkeit soll auch jeweils vor und nach der Migration eines Aspektes bestimmt und verglichen werden. Hier erwarten wir eigentlich eine Verbesserung. |
| 8 | Testbarkeit nimmt ab | Hier soll die Testabdeckung durch JUnit-Tests geprüft und verglichen werden. Tendenziell erwarten wir auch hier testbareren Code |
| 9 | Look and Feel wird von Anwendern nicht toleriert | Das „Look and feel“ soll vor und nach der Migration identisch sein, oder wird mit der Usability-Expertin abgestimmt werden. |

Vorbereitung

* 1. Architektur Eclipse 3 vs 4?

Organisatorisches

Projektstart

* 1. Ziele

Aspekt Iterationen

In den folgenden Kapiteln werden jeweils die einzelnen Aspekt-Iterationen beschrieben. Das Ziel ist es einen Überblick zu schaffen und die folgenden Fragen zu beantworten:

* Was wurde in der Iteration erreicht?
* Welche Probleme gab es?
* Was bringt der jeweilige Aspekt mit der E4 Lösung für Vorteile oder auch Nachteile?

Detaillierte Information über die Migrationsmöglichkeiten sind im Projekthandbuch zu finden.

* 1. Aspekt „Mixing E3/E4

In dieser Iteration wurden die diversen Möglichkeiten E3 und E4 Komponenten im selben Projekt zu mixen. Dies – also das Mixing von E3 und E4 - ist für eine schrittweise Migration einer grossen Anwendung, wie RCS es ist, eine zwingende Voraussetzung.

* + 1. Definition Abnahmekriterien

Die Abnahmekriterien wurden vom Betreuer folgendermassen definiert:

1. Möglichkeiten aufgeführt und kritisch bewertet und beschrieben
2. Prototyp und Demo
   * 1. Vergleich mit der E3 und der E4 Lösung

Die Vorteile von E4 gegenüber E3 sind:

* Moderneres Programmiermodell
* Vielfältige APIs aus E3 wurden deutlich reduziert und vereinheitlicht
* Singletons wurden entfernt.
* Moderne Konzepte wie Dependency Injection und Declarative Styling
  + 1. Einschränkungen und Risiken

Die folgenden Risiken bei einer E4 Einführung sind zu berücksichtigen:

* Performance bei Eclipse 4.2 noch nicht befriedigend
* Stabilität bei Eclipse 4.2 noch nicht befriedigend
* APIs noch nicht final, evtl. gibt es hier noch Anpassungen
  + 1. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3
    2. Migrationsmöglichkeiten
    3. Erfahrungen

Probleme mit Versionen

Probleme mit veralteten teilweise ungenauen Beschreibungen

In den Büchern sind viele Mickey Mouse Beispiele, aber die richtig harten Dinge werden nicht behandelt (z.B. Fragment nur halbwegs erklärt)

Problem mit E4.3 soll man das jetzt bereits nehmen oder nicht.

Mit Compability Layer funktioniert nicht alles 🡪 Das sei aber kein Fehler von E4, es war eher Zufall dass das vorer (E3) funktioniert hat.

Teilweise seltsame, unverständliche Fehlermeldungen

Wenn nur mit Compability Layer  QuickSearch links, mit LegacyIDE.xmi rechts

Wenn MWindow anstatt MTrimmedWindow --> Problem: Unable to process "WorkbenchWindow.model": no actual value was found for the argument "MTrimmedWindow".

Mit DiViewPart  Versionen

Mit LegacyIDE.xmi  Mehrere Hauptfenster gehen auf!?

Mit Compability Layer : Menüpunkte heissen plötzlich anders..

In Workspace Data Location wird tatsächlich das workbench.xmi der Application abgelegt…

* + 1. Fazit
  1. Aspekt „Adapter / Dependency Injection

Mit diesem Aspekt wurden das Thema Adapter und Dependency Injection behandelt. Bei den Adaptern war das Ziel die Migration der E3 Adapter nach E4. Mit dem Thema Dependency Injection sollte erörtert werden was dieser Mechanismus alles bietet und wie man die bestehenden Services und andere Framework-Komponenten durch die E4 Services ersetzt.

* + 1. Definition Abnahmekriterien

Die Abnahmekriterien wurden vom Betreuer folgendermassen definiert:

* Eine beispielhafte Implementierung von E4 Adapter Factories (falls vorhanden) und eine beispielhafte Nutzung eines Adapters (ch.sbb.rcsd.client.map.ui.IMappable) mit E4 Mitteln
* Eine beispielhafte Deklaration eines (eigenen) Services, z.B IAuthenticationService und eine beispielhafte Nutzung dieses Services (Anzeige des aktuellen Benutzers)
* Beschreibung des Aspektes im Handbuch
* Aktualisierter Projektbericht
  + 1. Vergleich mit der E3 und der E4 Lösung
    2. Einschränkungen und Risiken
    3. Qualität und Testbarkeit im Vergleich zu Eclipse RCP 3
    4. Migrationsmöglichkeiten
    5. Erfahrungen

Probleme

http://wiki.eclipse.org/Eclipse4/RCP/EAS/List\_of\_All\_Provided\_Services

Wenn field mit @Inject annotiert wird:

org.eclipse.e4.core.di.InjectionException: Unable to process "ZwlViewPart.site": no actual value was found for the argument "IWorkbenchPartSite".

Wenn in @PostConstruct angegeben:

Keine Fehlermeldung, aber Methode wird nicht aufgerufen

* + 1. Fazit

1. Reflexion