**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: | 0.3 |
| 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 |
| 12.07.2013 | 0.3 | MIRO | Aktualisierung Berichte für die ersten beiden Aspekte |
| 31.07.2013 | 0.4 | MIRO | Aspekt 3 Commands / Handler, Menus, Key Bindings |

Inhaltsverzeichnis

1. Einleitung 5

1.1. Zweck des Dokumentes 5

2. Projekt / Projektziele 5

2.1. Anlass und Begründung des Projektes 5

2.2. Problemstellung 6

2.3. Randbedingungen 6

2.4. Situationsanalyse 6

2.5. Erbrachte Vorleistung 6

2.6. Stakeholder 6

2.7. Zielvorstellungen 6

2.8. Lösungen 7

2.9. Sicherheits- und Datenschutzaspekte 7

3. Risiken 8

3.1. Risikoidentifizierung, -bewertung und -quantifizierung 8

3.2. Risikobehandlung 9

4. Vorbereitung 10

4.1. Know-how Aufbau 10

5. Organisatorisches 10

6. Projektplan 10

7. Projektstart 10

7.1. Ziele 10

8. Aspektermittlung 10

9. Aspekt Iterationen 11

9.1. Aspekt „Mixing E3/E4“ 11

9.1.1. Definition Abnahmekriterien 11

9.1.2. Dauer der Iteration 11

9.1.3. Probleme 11

9.1.4. Erfahrungen 12

9.1.1. Massnahmen 12

9.1.2. Risiken 13

9.1.1. Lieferobjekte 13

9.1.2. Fazit 13

9.2. Aspekt „Adapter / Dependency Injection“ 13

9.2.1. Definition Abnahmekriterien 13

9.2.1. Dauer der Iteration 13

9.2.2. Probleme 14

9.2.3. Erfahrungen 14

9.2.4. Risiken 14

9.2.5. Massnahmen 14

9.2.1. Lieferobjekte 14

9.2.2. Fazit 15

9.3. Aspekt „Commands / Handler, Menus, Key Bindings“ 15

9.3.1. Definition Abnahmekriterien 15

9.3.2. Dauer der Iteration 15

9.3.3. Probleme 15

9.3.4. Erfahrungen 16

9.3.5. Risiken 16

9.3.6. Massnahmen 16

9.3.7. Lieferobjekte 16

9.3.8. Fazit 16

10. Reflexion 16

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. Know-how Aufbau

Da ich im aktuellen Projekt täglich mit dem Eclipse 3 SDK arbeite und bereits einige UIs programmiert habe ist mein Know-How in diesem Bereich bereits solide. Eclipse 4 war mir bis auf einen Kursbesuch relativ unbekannt. Deshalb habe ich mich im Vorfeld der Masterarbeit in die Thematik Eclipse 4 eingearbeitet. Dies geschah hauptsächlich mit der Lektüre der beiden Eclipse 4 Bücher

|  |  |
| --- | --- |
| Eclipse 4 - Rich Clients mit dem Eclipse 4.2 SDK | Teufel, Marc, Helming, Jonas |
| Eclipse 4 Application Development | Vogel, Lars |

Organisatorisches

Projektplan

Grafik mit Phasen

Projektstart

* 1. Ziele

Aspektermittlung

Bei der Ermittlung der Aspekte haben der Mike und der Betreuer in drei Sessions diskutiert welche Aspekte allgemein aber auch spezifisch für RCS sinnvoll wären.

TODO komplette Aspektermittlung auflisten

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:

* Wie lange hat die Iteration gedauert?
* 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 jeweils im Projekthandbuch zu finden.

* 1. Aspekt „Mixing E3/E4“

In dieser Iteration wurden die diversen Möglichkeiten E3 und E4 Komponenten in derselben Applikation 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. Dauer der Iteration

Die Iteration war in der Periode

03.06.2013 – 16.06.2013

geplant, tatsächlich gedauert hat sie die Periode

03.06.2013 – 19.06.2013

gedauert. Die geplante Zeit wurde somit 3 Tage überschritten. Weil dieser Aspekt die Grundlage für die weiteren Aspekte bildet, wurde das toleriert. Die Zeitüberschreitung ist zum einen durch die Unterschätzung des Umfangs dieses Aspektes und zum anderen durch die unvorhergesehenen Probleme zu begründen. Die aufgetretenen Probleme dieser Iteration sind im nächsten Kapitel aufgeführt.

* + 1. Probleme

In diesem Kapitel wird beschrieben welche Probleme aufgetreten sind die das Voranschreiten der Arbeiten behindert hat.

Versionsprobleme Targetplattform

Bei der Implementierung in RCS kam es zu Problemen mit den Versionen. Die gewählte Targetplattform und die E4 Tools welche die 3.x e4-Bridge beinhaltet waren zuerst nicht kompatibel. Die Suche nach der richtigen Version hat sich als mühsam erwiesen. Es ging eine Weile bis ich überhaupt eine Seite für den Download der Version gefunden habe. Die Links auf der Seite <http://wiki.eclipse.org/E4/Install>

erwiesen sich immer wieder als veraltet. Die Übersicht auf <http://download.eclipse.org/e4/downloads/>

war da schon eher hilfreich.

Veraltete, falsche und ungenaue Beschreibungen

Viele Quellen im Internet erwiesen sich als veraltet und teilweise auch falsch. So ist zum Beispiel auf der Seite <http://www.vogella.com/articles/Eclipse4MigrationGuide/article.html> von Vogel Lars beschrieben, dass die Datei LegacyIDE.e4xmi – welche für eine mögliche Migrationsart vonnöten ist - im Plugin org.eclipse.ui.workbench zu finden sei. Ich habe dieses File schliesslich im Plugin org.eclipse.platformgefunden.

Teilweise wurden unbedingt benötigte Schritte eines Beispiels völlig weggelassen.

Keine Versionangaben in Quellen

Leider war den meisten Quellen nicht zu entnehmen für welche Version von Eclipse das jeweilige Beispiel erstellt wurde. Dieses Problem würde vielleicht das obenstehende Problem (Veraltete und falsche Beschreibungen) lösen. Mit einer Versionsangabe würde der Leser die beschriebenen Schritte kritischer betrachten und sich selber die Frage stellen, ob diese Lösung auch für seine Version von Eclipse in Frage kommt.

0815 Beispiele

Die meisten Beispiele im Internet wie auch in den E4 Bücher sind sehr kleine, beschränkte Beispiele. So habe ich zum Beispiel nirgends eine brauchbare und vollständige Anweisung gefunden wie man ein dem Application Model ein Window über ein Fragment hinzufügen kann. Da bedeutete teilweise, dass man Neuland betreten und Versuche starten musste.

Seltsame und unverständliche Fehlermeldungen

Bei der Migrationsmöglichkeit „LegacyIDE.e4xmi mit Fragment“ habe ich dem Application Model beim ersten Versuch ein Window also MWindow hinzugefügt. Dies führte zum folgenden seltsamen Fehler: Unable to process "WorkbenchWindow.model": no actual value was found for the argument "MTrimmedWindow". Danach habe ich es mit einem TrimmedWindow also MTrimmedWindow probiert und es gab keine Exception mehr. Genau das gleiche hat mit dem Ansatz „LegacyIDE.e4xmi mit Prozessor“ ohne Probleme geklappt.

Suche nach geeignetem Beispiel in RCS

Viele Views in RCS sind sehr eng an Eclipse 3 APIs gekoppelt, somit war es schwierig ein geeignetes Beispiel zu finden um auschliesslich den Aspekt „Mixing E3 und E4“ zu behandeln.

* + 1. Erfahrungen

Nicht alles funktioniert einwandfrei

Die E3 Applikation läuft mit dem Compability Layer und dem LegacyIDE.e4xmi Ansatz. Es muss jedoch erwähnt werden, dass teilweise Menüeinträge erscheinen die auf der E3 Targetplattform nicht da sind. Beim Hauptfenster befindet sich zum Beispiel neu der Menüpunkt Ansicht in der Toolbar. Die Stellungnahme hierzu von Eclipse: Dies sein kein Fehler von E4 es sei eher Zufall dass das vorher in E3 funktioniert hat. Hier muss also nachkorrigiert werden.

Quick Access nicht optional

Mit E4 ist der „Quick Access“ sichtbar, es kann nicht verstecckt werden. Dies könnte sich für die RCS-Anwender als störend erweisen. Der Bug ist unter <https://bugs.eclipse.org/bugs/show_bug.cgi?id=362420>

beschrieben.

Unterschiedliches Verhalten von Ansatz mit Prozessor und Fragment

Beim Ansatz mit Prozessor öffnen sich wie erwartet zwei Fenster, beim Fragment -Ansatz öffnen sich seltsamerweise deren drei. Dies obwohl sich die beiden Ansätze identisch verhalten sollten. „Quick Access“ ist bei ein paar Fenster links von der Toolbar eingeordnet, bei anderen Fenstern rechts davon.

* + 1. Massnahmen
    2. Risiken

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

* + 1. Lieferobjekte

Es wurden diverse Wege für das Zusammenspiel von E3 und E4 Plugins aufgezeigt und im Handbuch schrittweise beschrieben. Die für interessant erklärten Wege wurden in RCS ausimplementiert.

TODO Was noch?

* + 1. Fazit

Es wurden - trotz den Problemen - für RCS durchaus gangbare Wege zur Migration aufgezeigt und detailliert beschrieben.

* 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. Dauer der Iteration

Die Iteration war in der Periode

17.06.2013 – 30.06.2013

geplant, tatsächlich gedauert hat sie die Periode

20.06.2013 – 07.07.2013

gedauert. Der spätere Beginn ist auf die Verzögerung in der vorherigen Iteration zurückzuführen. Die geplante Zeit wurde somit 7 Tage überschritten. Diese Überschreitung ist nicht tolerierbar und muss genauer erklärt werden. Die folgenden drei Hauptgründe sind für diese Verspätung verantwortlich:

1. Versionssprung von Targetplattform 4.2 nach 4.3
2. Thema umfangreicher als angenommen
3. Verlängertes Wochenende
4. Krankheit

Ich habe mich dazu entschlossen von Version 4.2 auf 4.3 zu gehen um von den Stabilisierungen und den Verbesserungen in der Performance zu profitieren. Ein weiterer Grund für den Versionssprung war schlichtweg auf dem aktuellen Stand zu sein. Das Ganze hat zu einigen Verzögerungen der Arbeiten geführt, die Probleme sind dem nächsten Kapitel zu entnehmen. Ich habe diesen Versionswechsel absolut unterschätzt.

Ein verlängertes Ferienwochenende meinerseits vom 20.6. bis 23.6. hat zu einem unerwarteten Bruch des Arbeitsflusses geführt.

Ich war vom 5.7. bis 7.7. krankheitshalber angeschlagen, die Produktivität hat sehr darunter gelitten.

Das Thema Dependency Injection bietet viel mehr als ursprünglich angenommen, man kommt bei diesem Thema zum Beispiel nicht an den Services vorbei.

* + 1. Probleme

In diesem Kapitel wird beschrieben welche Probleme aufgetreten sind die das Voranschreiten der Arbeiten behindert hat.

Evaluation Versionssprung 4.3

Ich habe mich bei der Evaluation ob wir auf 4.3 wechseln sollen oder nicht ziemlich verzettelt. Konkret habe zu lange Zeit mit der Suche nach Beiträgen über Vorteile von E 4.3 gegenüber 4.2 verbracht.

Versionssprung 4.3

Ich habe mich nach der Lektüre der Vorteile wie zum Beispiel hier <http://www.heise.de/developer/meldung/Deutlich-bessere-Performance-bei-Eclipse-4-3-1751323.html> oder <http://download.eclipse.org/eclipse/downloads/drops4/R-4.3-201306052000/news/> dazu entschlossen auf E4.3 zu wechseln. Dieser Entschluss hat zu ungeahnten Aufwänden bei der Migration von 4.2 zu 4.3 geführt.

Beim ersten Versuch auf 4.3 zu wechseln gab es noch keine passende Version der E4 Tools, das führte dazu, dass die Migration mittels E4 Bridge nicht mehr funktionierte. Dies bedeutet einen Schritt zurück zu treten, und zwar zur Version 4.2. Zwei Wochen später gab es dann die gewünschte Version.

Ein weiteres Problem gab es mit dem Plugin org.junit bzw. org.junit4. Mit dem Kepler Bundle ist org.junit4 nicht mehr brauchbar. Leider waren alle Client Test-Plugins von genau diesem Plugin org.junit4 abhängig. Diese Abhängigkeit musste in allen Plugins auf org.junit geändert werden.

* + 1. Erfahrungen

Nicht alles funktioniert einwandfrei mit Targetplattform 4.3

Wenn man im Eclipse Juno, also 4.2, probiert die Targetplattform auf 4.3 zu wechseln gibt es die Fehlermeldung „You have selected a target with a newer version than your current Eclipse installation. This can cause unexpected behaviour in PDE. Please use a newer version of eclipse“. Es könnte also sein, dass diese Kombination zu unerwartetem Verhalten führen kann. Mit Eclipse 4.3 habe ich jedoch aktuelle Probleme mit Subversion. Das Subversion Plugin erkennt die Eclipse-Projekte nicht als von Subversion versionierte Projekte.

Dependency Injection

Es gab inkonsistentes Verhalten beim Injizieren von nicht injizierbaren Klassen. Der Versuch eine WorkbenchPartSite auf einem POJO zu injizieren verhält sich bei @Inject annotierten Felder ander als bei mit @PostConstruct annotierten Methoden.

Die @Inject Variante führt zu folgender Fehlermeldung: org.eclipse.e4.core.di.InjectionException: Unable to process "ZwlViewPart.site": no actual value was found for the argument "IWorkbenchPartSite".

Bei der @PostConstruct Variante wird die Methode einfach stillschweigend **nicht** aufgerufen.

* + 1. Risiken
    2. Massnahmen

Keine andere Arbeiten mehr als geplant, z.B. Versionssprünge etc..

* + 1. Lieferobjekte

Mit dieser Iteration wurden die Anleitung und ein konkretes Beispiel wie man Adapters von E3 nach E4 migrieren kann geliefert. Es wurde das Thema Dependency Injection ausführlich erklärt und beschrieben. Auch hier wurde ein konkretes Beispiel implementiert. TODO

* + 1. Fazit

Die Iteration kann, obwohl, sie deutlich länger als geplant gedauert hat, als erfolgreich betrachtet und abgeschlossen werden.

* 1. Aspekt „Commands / Handler, Menus, Key Bindings“

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. Dauer der Iteration

Die Iteration war in der Periode

01.07.2013 – 14.06.2013

geplant, tatsächlich gedauert hat sie vom

08.07.2013 – 31.07.2013

gedauert. Der spätere Beginn ist auf die Verzögerung in den vorherigen Iterationen zurückzuführen. Die geplante Zeit wurde somit 7 Tage überschritten. Diese Überschreitung ist nicht tolerierbar und muss genauer erklärt werden. Die folgenden drei Hauptgründe sind für diese Verspätung verantwortlich:

1. Versionssprung von Targetplattform 4.2 nach 4.3
2. Verlängertes Wochenende
3. Krankheit
4. Thema umfangreicher als angenommen

Ich habe mich dazu entschlossen von Version 4.2 auf 4.3 zu gehen um von den Stabilisierungen und den Verbesserungen in der Performance zu profitieren. Ein weiterer Grund für den Versionssprung war schlichtweg auf dem aktuellen Stand zu sein. Das Ganze hat zu einigen Verzögerungen der Arbeiten geführt, die Probleme sind dem nächsten Kapitel zu entnehmen. Ich habe diesen Versionswechsel absolut unterschätzt.

Ein verlängertes Ferienwochenende meinerseits vom 20.6. bis 23.6. hat zu einem unerwarteten Bruch des Arbeitsflusses geführt.

Ich war vom 5.7. bis 7.7. krankheitshalber angeschlagen, die Produktivität hat sehr darunter gelitten.

Das Thema Dependency Injection bietet viel mehr als ursprünglich angenommen, man kommt bei diesem Thema zum Beispiel nicht an den Services vorbei.

* + 1. Probleme

In diesem Kapitel wird beschrieben welche Probleme aufgetreten sind die das Voranschreiten der Arbeiten behindert hat.

E3 Action, Commands etc. zu wenig bekannt

Ich habe eindeutig unterschätzt wie vielfältig dieser Aspekt ist. Mir waren nicht alle Konzepte – vor allem im E3 Bereich - bekannt und deshalb war Einarbeitung nötig. Das hat mehr Zeit gekostet als geplant.

* + 1. Erfahrungen
    2. Risiken
    3. Massnahmen

Da die Zeit langsam knapp wird habe ich mit dem Betreuer beschlossen die nächsten zwei Wochen jeweils einen Tag mehr frei zu nehmen. Mit diesem zusätzlichen Tag pro Woche sollte es möglich sein die verlorene Zeit aufzuholen.

* + 1. Lieferobjekte
    2. Fazit

1. Reflexion