**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: | 0.72 |
| 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 |

**Versionskontrolle**

|  |  |  |  |
| --- | --- | --- | --- |
| **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“ |
| 04.08.2013 | 0.5 | MIRO | Kapitel Organisatorisches, Setup und Vorgaben ergänzt |
| 08.08.2013 | 0.6 | MIRO | Aspekt 4 „Eigene Extension Points / Eigene Services“, Termine |
| 14.08.2013 | 0.7 | MIRO | Aufräumarbeiten |
| 16.08.2013 | 0.71 | MIRO | Nachtrag Aspekt 3, Aufräumarbeiten |
| 04.09.2013 | 0.72 | MIRO | Nachtrag Aspekt ? |

Inhaltsverzeichnis

1. Einleitung 6

1.1. Zweck des Dokumentes 6

2. Projekt / Projektziele 6

2.1. Anlass und Begründung des Projektes 6

2.2. Problemstellung 7

2.3. Randbedingungen 7

2.4. Situationsanalyse 7

2.5. Stakeholder 7

2.6. Zielvorstellungen 8

2.7. Lösungen 8

2.8. Sicherheits- und Datenschutzaspekte 8

3. Risiken 9

3.1. Risikoidentifizierung, -bewertung und -quantifizierung 9

3.2. Risikobehandlung 10

4. Vorbereitung 11

4.1. Know-how Aufbau 11

5. Setup 11

6. Organisatorisches 12

7. Vorgaben 13

7.1. Beurteilungskriterien 13

7.2. Kapitelname? TODO 13

7.3. Tipps vom Experten 13

7.4. Checkliste Dokumente Master Thesis TODO 14

1. Titelblatt 14

2. Anforderungen Abgabe 14

8. Projektplan 15

9. Projektstart 17

9.1. Ziele 17

10. Aspekt Ermittlung 18

11. Aspekt Iterationen 20

11.1. Aspekt „Mixing E3/E4“ 20

11.1.1. Definition Abnahmekriterien 20

11.1.2. Dauer der Iteration 20

11.1.3. Resultate 21

11.1.4. Test 21

11.1.5. Probleme 22

11.1.6. Erfahrungen 23

11.1.1. Risikobeurteilung 23

11.1.1. Massnahmen 23

11.1.1. Fazit 23

11.2. Aspekt „Adapter / Dependency Injection“ 24

11.2.1. Definition Abnahmekriterien 24

11.2.1. Dauer der Iteration 24

11.2.1. Resultate 25

11.2.2. Test 26

11.2.3. Probleme 27

11.2.4. Erfahrungen 27

11.2.5. Risikobeurteilung 27

11.2.6. Massnahmen 27

11.2.7. Fazit 27

11.3. Aspekt „Commands / Handler, Menus, Key Bindings“ 28

11.3.1. Definition Abnahmekriterien 28

11.3.2. Dauer der Iteration 28

11.3.3. Probleme 28

11.3.4. Erfahrungen 30

11.3.5. Risiken 30

11.3.6. Massnahmen 30

11.3.1. Lieferobjekte 30

11.3.2. Fazit 30

11.4. Aspekt „Eigene Extension Points / Eigene Services“ 30

11.4.1. Definition Abnahmekriterien 30

11.4.2. Dauer der Iteration 30

11.4.3. Probleme 31

11.4.4. Erfahrungen 31

11.4.5. Risiken 31

11.4.6. Massnahmen 31

11.4.7. Lieferobjekte 31

11.4.8. Fazit 31

12. Aufräum-Iteration zwischen Aspekt 3 und 4 31

13. Zwischenreview 31

14. Aspekt Application Model vs. Advisors 31

15. Reflexion 32

16. Präsentation 32

Beurteilungskriterien für Diplompräsentationen 32

1. Publikumskontakt 32

2. Stimme 32

3. Präsentationsmittel 32

4. Aufbau / Gliederung 32

5. Zeitmanagement 32

6. Gesamtbild 32

17. Verzeichnisse / Quellen 32

17.1. Abbildungsverzeichnis 32

17.1. Quellverzeichnis 33

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*](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 werden 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. 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) |

Der Projektbetreuer Marc Hoffmann ist seit mehreren Jahren (TODO) im Projekt RCS in verschiedenen Teilprojekten tätig. Er hat als Entwickler und auch als Architekt ein enormes Wissen über RCS aufgebaut. Dazu hat er in den Jahren TODO bei Eclipse gearbeitet, er bringt als auch ein enormes Wissen über Eclipse und sonstige technische Themen mit.

Diese Punkte führen dazu, dass Marc der perfekte Projektbetreuer darstellt.

* 1. Zielvorstellungen

Das Ziel des Projektes ist es mindestens 4 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 Aspekte und deren Migration werden im Projekthandbuch dokumentiert. Die Erfahrungen im Projekt sollen in diesem 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 technischen 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 und der beiden E3 Bücher

|  |  |
| --- | --- |
| **E4 Bücher** |  |
| Eclipse 4 - Rich Clients mit dem Eclipse 4.2 SDK | Teufel, Marc, Helming, Jonas |
| Eclipse 4 Application Development | Vogel, Lars |
|  |  |
| **E3 Bücher** |  |
| Eclipse Rich Client Platform | Chris Aniszczyk, Jean-Michel Lemieux, Jeff McAffer |
| Eclipse RCP im Unternehmenseinsatz | Stefan Reichert |

Setup

Es wurde bestimmt die Dokumentation der Arbeit, also Projektbericht und auch das Projekthandbuch mittels Microsoft Office Tools zu erstellen. Der Sourcecode wird in Java Files gehalten.

Um die produzierten Dokumente und andere Erzeugnisse der Masterarbeit versioniert, gesichert und von überall zugänglich abzulegen wurde das Repository von Github gewählt.

Konkret finden sich die Erzeugnisse unter <https://github.com/MikeR13/MAS/>.

Github bietet noch ein anderes Feature, nämlich die Erfassung von Milestones und Issues. Dies soll in diesem Projekt auch zum Einsatz kommen. Es sollen jeweils Milestones mit bestimmten Zieldaten erfasst werden. Dies kann zum Beispiel die Bearbeitung eines Aspektes sein. Zu jedem Milestone sollen die benötigten Issues oder in diesem Kontext besser Tasks erfasst werden. Wenn ein Task erledigt ist, so soll er geschlossen werden, also den Status closed erhalten. TODO genauere Vorgaben für Issues etc.

Evtl. auch Commit Activity oder ähnliches aufzeigen

Der Sourcecode wird im SBB eigenen Subversion (SVN) Repository abgelegt. Dazu wird ein eigener Branch erstellt. Der Branch wird im Gegensatz zum Trunk gegen die Targetplattform E4 kompiliert.

Der Branch wird mittels täglichem und automatischem Merge vom Trunk in den Branch aktuell gehalten. Sollte der automatische Merge wegen Konflikten nicht klappen, so muss dies von Hand gemacht werden. Bei einem missglückten automatischen Merge werden die interessierten Personen per Mail informiert. Der automatische Merge kann auch über einen Jenkins-Job manuell angestossen werden.

Auch die Tests – in diesem Falle JUnit Tests – können automatisch ausgeführt werden. Dies passiert mindestens einmal täglich oder direkt nach einem Commit auf dem Branch. Auch wenn hier etwas schief läuft wird der Verursacher des Fehlers – bzw. der Commiter – per Mail benachrichtigt.

Das Subversion Repository wird selbstverständlich gesichert, es kann bei Bedarf ein Restore erfolgen.

Die URL des Subversion Repositories lautet:

<http://rcssvn1/svn/dispo>

Die URL vom Buildserver lautet:

<http://rcsinfra2:8080/hudson/>

Der Buildserver ist – obwohl dies die URL nicht vermuten lässt - ein Jenkins[[1]](#footnote-1)-Buildserver. Die Builds können unter der folgenden URL gestartet, verwaltet und angeschaut werden:

<http://rcsinfra2:8080/hudson/job/RCS-Client-dispo_client_e4/>

Unter der folgenden URL ist es möglich die Resultate der Merge Jobs von trunk zum E4 branch anzuschauen und bei Bedarf Merge-Jobs von Hand zu starten:

<http://rcsinfra2:8080/hudson/job/Shell-SvnMerge-dispo_client_e4/>

In der folgenden Abbildung sehen wir eine Übersicht über die letzten Builds auf dem E4 Branch:



Abbildung 1 E4 Branch Builds

Organisatorisches

Von der Schule wird vorgeschrieben mindestens die folgenden Meetings mit dem Experten und dem Betreuer durchzuführen:

* Kickoff
* Zwischenreview (mindestens 1)
* Schlussreview

Mit dem Experten wurde vereinbart alle 2-4 Wochen einen Statusbericht abzugeben. Dafür wird von der Schule ein Template[[2]](#footnote-2) zur Verfügung gestellt. Der Statusbericht wird per Mail an den Experten und den Betreuer versandt. Zusätzlich werden die Statusberichte im Repository unter <https://github.com/MikeR13/MAS/tree/master/Deliverables> abgelegt.

Vorgaben

* 1. Beurteilungskriterien

Die in diesem Kapitel aufgeführten Beurteilungskriterien wurden zum einen im Kickoff-Meeting bestimmt.

Es wird vor allem das methodische Vorgehen bewertet. An erster Stelle kommt die Abschlussarbeit und dann erst das Projekt bzw. die Arbeit für den Arbeitgeber. Am Schluss soll bewertet werden, ob die Ziele erreicht werden konnten.

Beurteilungsschema

Das folgende Beurteilungsschema wurde für die Arbeit definiert:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Gewicht** | **Max. Punktzahl** |
| Vorbereitung | Projektdefinition, Wahl der Vorgehensmethodik | 2 | 10 |
| Aufwandschätzung, Zeitplanung | 1 | 10 |
| Risiko-Erhebung | 2 | 10 |
| Teststrategie und Testkriterien | 2 | 10 |
|  | | | |
| Durchführung | Fachmethodik | 3 | 10 |
| Implementation | 1 | 10 |
| Anforderungsmanagement | 3 | 10 |
| Verifizierung, Test | 3 | 10 |
| Projektmanagement | 1 | 10 |
| Kreativität, Initiative, Selbstständigkeit | 2 | 10 |
| Risiko-Management | 2 | 10 |
|  |  |  |  |
| Ergebnis | Übereinstimmung Produkt/Anforderungen | 1 | 10 |
| Dokumentation | 4 | 10 |
| Abnahme | 2 | 10 |
| Nachvollziehbarkeit | 3 | 10 |

Gesamtbeurteilung

| **Note** | **Gewicht** |
| --- | --- |
| Fachliche Beurteilung | 90% |
| Präsentation | 10% |

* 1. Kapitelname? TODO

Es wird vom Experten kein/e Feedback/Empfehlungen zum aktuellen Verlauf des Projektes geben, selbständiges Arbeiten ist angesagt.

* 1. Tipps vom Experten

Die folgenden Tipps stammen aus einem vom Experten abgegeben Dokument.

Treten im Verlauf der Arbeit grössere Schwierigkeiten auf, die ein erfolgreiches Abschliessen der Arbeit gefährden, so sind der Betreuer und der Experte darüber zu informieren. Gemeinsam wird das weitere Vorgehen definiert und beschlossen.

Führen Sie ein Tagebuch, in dem Sie die wichtigsten Ereignisse und Notizen festhalten. Mit diesen Notizen fällt Ihnen das Schreiben des Berichtes viel leichter.

Erstellen Sie eine Checkliste der Ergebnisse, die Sie erarbeiten oder dokumentieren wollen, und legen Sie fest, in welches Lieferobjekt sie zu liegen kommen.

Führen Sie im Bericht nur Referenzen auf, die im Text verwendet werden und geben Sie die Quelle an. Ist die Quelle für mich als Experte nicht zugänglich, so geben Sie sie als Beilage der Arbeit mit oder zitieren Sie die notwendigen Passagen dazu.

Und zum Schluss:

„Was nicht dokumentiert ist, existiert nicht, und was nicht existiert, kann nicht bewertet werden.

TODO

Bericht im Umfang von 30 – 50 Seiten mit Informationen was alles im Projekt vorgegangen ist, Risiken (Erhebung, Beurteilung, …), Testing (Wie testen, erwartete Testresultate, Ergebnisprüfung), Methodik/Projektmanagement (Planen, tracken, Retro, Bewerten), Lessons learned

Nur Referenzen erwähnen die auch im Bericht vorkommen

Wenn Quelle nicht verfügbar 🡪 kleines Abstract

Was nicht dokumentiert ist, existiert nicht.

Artefakt 🡪 wie testen? 🡪 Test durchführen

* 1. Checkliste Dokumente Master Thesis TODO

## Titelblatt

* SWS-Logo ([svg](http://www.sws.bfh.ch/gfx/bfhlogo+schrift.svg), [png](http://www.sws.bfh.ch/gfx/bfhlogo+schrift.png))
* Nummer der Abschlussarbeit (z.B. MT-10-01.09)
* Titel (Titel gegenüber der Themeneingabe nicht mehr ändern!)
* Klasse, Datum
* Abstract: Zusammenfassung der Master Thesis (Max. 300 Zeichen)
* Schlüsselwörter
* Name, Adresse und Tel. des Studenten/der Studentin
* Name, Adresse und Tel. des Betreuers/der Betreuerin
* Name des Experten/der Expertin

## Anforderungen Abgabe

* Abgabe (Upload) des Berichts und eventuellen Zusätzen (z.B. Sourcecode) über die Diplomplattform. Der Experte und der/die Betreuer werden dabei automatisch informiert.

Projektplan

In der folgenden Tabelle sind alle Termine aufgelistet die im Rahmen dieses Projektes anfallen bzw. angefallen sind. Bei den Einträgen in blauer Schrift handelt es sich um die Termine die von Anfang von der Schule festgelegt wurden. Die anderen Termine wurden im Laufe des Projektes ausgemacht.

|  |  |  |
| --- | --- | --- |
| **Datum** | **Termin** | **Beteiligte Personen** |
| 02.05.2013 | Kickoff Meeting | BU, HM, RM[[3]](#footnote-3) |
| 21.05.2013 | Abstract | HM, RM |
| 21.05.2013 | Projektantrag | HM, RM |
| 07.06.2013 | Statusbericht | BU, HM, RM |
| 24.06.2013 | Statusbericht | BU, HM, RM |
| 10.07.2013 | Zwischenreview | BU, HM, RM |
| 02.08.2013 | Statusbericht | BU, HM, RM |
| 16.08.2013 | Statusbericht | BU, HM, RM |
| 05.09.2013 | Schlussreview | BU, HM, RM |
| 12.09.2013 | Eingabe Präsentationsbedürfnisse | HM, RM |
| 12.09.2013 | Abgabe Dokumentation/Anhänge | HM, RM |
| 12.09.2013 | Upload Poster | HM, RM |
| 20.09.2013 | Schlusspräsentation | BU ([[4]](#footnote-4)), HM, RM |

TODO Termine mit Marc, die anderen Daten raussuchen!

15.5. MAS Abstract

8.8. 1.5 h Marc

16.8. Marc

29.08.2013 Marc

03.09.2013 Marc

Aufgrund der Termine, der geplanten Ferien und der anstehenden Arbeiten entstand die folgende Grobplanung für die Arbeiten:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **3 Wochen** | **12 Wochen** | **1 Woche** | **2 Wochen** | **2 Wochen** |
| **28.4. – 19.5.** | **20.5. – 11.8.** | **12.8. – 18.8.** | **19.8. – 1.9.** | **2.9. – 15.9.** |
| Vorbereitungen  Beurteilung wie was umgebaut werden kann  Aspekte ermitteln | Aspekte angehen, Analyse, Design Umsetzung | Abschluss | Ferien | Abschluss |

In den ersten 3 Wochen sollen die nötigen Vorbereitungen getroffen werden. Es soll klar werden welche Aspekte im Rahmen der Arbeit angegangen werden sollen. In den darauffolgenden 12 Wochen sollen die einzelnen Aspekte – einer nach dem anderen – bearbeitet werden. Die letzten knapp 3 Wochen (Die Ferien wurden bereits abgezogen) soll die Arbeit abgeschlossen werden Dies bedeutet, dass alle Dokumente, sämtlicher Code und sonstige Artefakte komplett und vom Betreuer abgenommen sind.

TODO Excel Tabelle mit geplanten und geleisteten Arbeiten hier reinkopieren

TODO Planung und tatsächliche Iterationen gegeneinander aufstellen

Projektstart

* 1. Ziele

Aspekt Ermittlung

Bei der Ermittlung der Aspekte hat der Autor zuerst alleine und später mit dem Betreuer der Arbeit in drei Sessions bearbeitet und diskutiert welche Aspekte allgemein aber auch spezifisch für RCS sinnvoll wären.

Bei einem Brainstorming wurden die folgenden möglichen Aspekte erwähnt:

* Selection
* Plugin.xml / Application.e4xmi
  + Verschiedene Aspekte darin..
  + Menu / MenuItem
  + Key Binding
  + Command, Handler, Action
  + Toolbar Contributions
  + Trim Contributions
  + Contributions bei E3 in diversen Registries (ViewReg., EditorReg.,..))
* Teile der Anwendung dynamisch erzeugen in E3 nicht konsistent und durchgängig (@see Kapitel 4.1.1, View, Editor, Action, Command, Perspective, ActionSet, MenuContributions, ActionBarAdvisor, WorkbenchAdvisor, WorkbenchWindowAdvisor, IPerspectiveFactory) --> neu alles über ApplicationModell möglich.
* Part vs. View/Editor
* Statische APIs vs. Context / E4 Services (SelectionService, Preferences, Model Service, Part Service, EventBroker, Translation Service, Eigene OSGi Services)
* CSS?
* Dependency Injection
  + IPreferenceStore s = IDEWorkbenchPlugin.getDefault().getPreferenceStore(); store.getBoolean("uhu"); vs. @Inject @Preference(uhu) boolean uhu;
* Eventhandling / Event broker
* Adapters
* Workbench Themes
* weg vom Workbench Modell?
* IEclipsePreferences
* Extension Points --> Fragments/Processors
* 3.x e4 bridge (Views etc. vorbereiten)
* Ersetzen Parts, z.B. Subclassing von ViewPart
* NLS
* Model Addons (CommandServiceAddon, ContextServiceAddon, BindingServiceAddon, CommandProcessingAddon, ContextProcessingAddons, BindingProcessingAddon, org.eclipse.e4.ui.workbenchaddons.swt für DnD --> DnDAddon und CleanupAddon..)
* SelectionDialogs in Eclipse 3, aber nicht in Eclipse 4 (@see BUCH Vogel in Mitte)
* IProgressMonitor

Die Dokumente zum Brainstorming und sonstige Notizen sind unter <https://github.com/MikeR13/MAS/tree/master/Unterlagen_Notizen/Brainstrorming_Aspekte>

abgelegt.

Da der Betreuer auch den Auftraggeber vertritt konnte er bestimmen welche Aspekte behandelt werden sollten. Folgende Aspekte wurden bestimmt:

|  |  |  |
| --- | --- | --- |
| **Aspekt Nummer** | **Name** | **Beschreibung** |
| 1 | Mixing E3 / E4 | Mit diesem Aspekt soll geprüft werden welche Migrationsmöglichkeiten grundsätzlich existieren. Können Eclipse RCP 3 und Eclipse RCP 4 Komponenten im selben Projekt gleichzeitig nebeneinander im Einsatz sein? |
| 2 | Adapters / Dependency Injection | Mit diesem Aspekt wird die Migration von den Adapter’s und Dependency Injection behandelt. Dependency Injection gibt es in E3 nicht, deshalb wird hier verglichen welche E4 Services den „alten“ E3 Services entsprechen. Die neuen Services können mittels Dependency Injection injiziert werden |
| 3 | Commands / Handler, Menus, Key Bindings | Mit diesem Aspekt sollen Commands, Handlers, Menus und auch das Key-Binding - also Shortcuts – behandelt werden. Wie sehen diese in E4 aus, wie in E3 und wie können sie von E3 nach E4 migriert werden. |
| 4 | Eigene Extension Points / Eigene Services | In diesem Aspekt geht es darum aufzuzeigen wie eigene Extension Points aus E3 in E4 abgelöst werden können. Kann dies eventuell über eigene Services geschehen? |
| 5 | Application Model vs. Advisors | Mit diesem Aspekt soll der Unterschied vom Application Model zu den E3 Advisors aufgezeigt werden. |

Die Auswahl der Aspekte ist nicht final das heisst, dass bei Bedarf ein Aspekt vom Betreuer neu bestimmt werden kann. Sobald aber ein Aspekt in Angriff genommen wird, soll aber möglichst nichts mehr geändert werden.

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?
* Testing
* Welche Probleme gab es?
* Risikobeurteilung
* Was bringt der jeweilige Aspekt mit der E4 Lösung für Vorteile oder auch Nachteile?

Detaillierte Information über die Migrationsmöglichkeiten und die Beantwortung von einzelnen Fragen sind jeweils im Projekthandbuch[[5]](#footnote-5) 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 in einem der nächsten Kapitel aufgeführt.

* + 1. Resultate

In diesem Kapitel sollen kurz die Resultate aus der Aspekt-Bearbeitung aufgezeigt werden. Die Resultate und der Weg zu diesen Resultaten sowie die ausführliche Dokumentation zu diesem Aspekt sind im Projekthandbuch[[6]](#footnote-6) im Kapitel Aspekt „Mixing E3 / E4“ zu finden.

Was wurde erreicht?

Es konnten diverse Wege aufgezeigt werden, wie in einer Eclipse RCP Anwendung E3 und E4 Komponenten zusammen gemixt werden können. Dies wurde theoretisch wie auch praktisch erklärt und durchgeführt. Die für interessant erklärten Wege wurden in RCS ausimplementiert.

TODO Was noch?

Gegenüberstellung E3 und E4

In der folgenden Tabelle werden diverse Themen zu „Mixing E3/E4“ einander gegenübergestellt. Die Tabelle soll einen Überblick schaffen wo sich E3 und E4 unterscheiden oder wo sie gleich sind.

|  |  |  |
| --- | --- | --- |
| **Thema** | **E3** | **E4** |
| Starten der Applikation | plugin.xml, WorkbenchAdvisor, WorkbenchWindowAdvisor, ActionBarAdvisor | plugin.xml, Application Model |
| Parts erstellen | plugin.xml, Erben von ViewPart | Application Model, POJO |
| Deklaration von Parts, Menus, etc. | plugin.xml im Application Plugin und plugin.xml in weiteren Plugins | plugin.xml und Application Model (z.B. application.e4xmi oder LegacyIDE.e4xmi) im Application Plugin und plugin.xml und fragment.e4xmi in weiteren Plugins. Möglichkeit von Prozessoren. |
| Werkzeug für Deklaration | plugin.xml Editor | Application Model Editor |
| Modeled UI | Nein | Ja |
| Dependency Injection | Nein | Ja |
| CSS | Nein | Ja |
| SWT/JFace | Ja | Ja, es besteht aber Möglichkeit über andere Renderer zum Beispiel JavaFX einzubinden |
| Runtime | Equinox, JVM | Equinox, JVM (ab Version 6) |
| TODO mehr? Evtl. Compability Layer? |  |  |

Beurteilung Migration

Die Migration scheint mit den zur Verfügung stehenden Mitteln durchaus machbar. Es stehen diverse Möglichkeiten zur Verfügung. Es wird sich bei der Bearbeitung der nächsten Aspekte herausstellen, ob diese Vermutung richtig ist.

* + 1. Test

Getestet wurde dieser Aspekt ausschliesslich über das Starten der Applikation. Es hat nach anfänglichen Schwierigkeiten alles geklappt.

TODO Für Stabilität: Applikation laufen lassen, ein paar Kicks

TODO Beweisbilder fürs Testing

* + 1. Probleme

In diesem Kapitel wird beschrieben welche Probleme aufgetreten sind die das Voranschreiten der Arbeiten behindert hat. Es sollen auch Probleme beschrieben die im Projektalltag aufgetreten sind, also nicht nur technische Probleme.

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 Versionsangaben 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 ausschliesslich 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 versteckt 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 Fenstern links von der Toolbar eingeordnet, bei anderen Fenstern rechts davon.

* + 1. Risikobeurteilung

|  |  |  |
| --- | --- | --- |
| **Id** | **Risiko** | **Risikobehandlung** |
| 1 | Migration wegen unbekannten technischen Problemen nicht möglich | Die Einschätzung dieses Risikos hat sich nicht verändert |
| 2 | Zeitplan kann nicht eingehalten werden | Der Zeitplan wurde so nicht eingehalten, er wurde um 3 Tage überschritten. Dies wurde jedoch toleriert und wird in den nächsten Aspekten wieder eingehalten |
| 3 | Funktionalität geht verloren | Mit den bisherigen Migrationsschritten konnte kein Verlust an Funktionalität festgestellt werden. |
| 4 | Performanceeinbussen | Das Eintreten dieses Risikos wurde nicht überprüft. Es soll am Ende |
| 5 | Verlust von Stabilität | Es konnte kein Verlust an Stabilität festgestellt werden. |
| 6 | Verlust von Usability | Es konnte kein Verlust von Usability festgestellt werden. |
| 7 | Wartbarkeit nimmt ab | Mit dem Ansatz vom Application Model ist die Wartbarkeit eher gestiegen |
| 8 | Testbarkeit nimmt ab | Mit dem Ansatz vom Application Model ist die Wartbarkeit eher gestiegen |
| 9 | Look and Feel wird von Anwendern nicht toleriert | Das Look and feel hat keine Änderungen erfahren |

Weitere, neue Risiken konnten nicht festgestellt werden.

* + 1. Massnahmen

Da die Überschreitung der Iterationszeit nicht als katastrophal einzustufen ist und sonst die Ziele erreicht wurden keine Massnahmen definiert. Aus Risikosicht hat sich auch nicht viel geändert.

Aspekt 2 kann mit gutem Gewissen in Angriff genommen werden.

* + 1. Fazit

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

Lessons learned

* 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 4 Tage überschritten. Diese Überschreitung ist nicht tolerierbar und muss genauer erklärt werden. TODO Das zu den Problemen nehmen?: 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. Resultate

In diesem Kapitel sollen kurz die Resultate aus der Aspekt-Bearbeitung aufgezeigt werden. Die Resultate und der Weg zu diesen Resultaten sowie die ausführliche Dokumentation zu diesem Aspekt sind im Projekthandbuch[[7]](#footnote-7) in den Kapiteln „Aspekt Adapter“ und „Aspekt Dependency Injection“ zu finden. Es handelt sich also eigentlich um zwei Aspekte. Deswegen werden die Resultate auch separat aufgeführt.

Dependency Injection

Was wurde erreicht?

Es konnte ein Weg aufgezeigt werden, wie die statischen Methodenaufrufe inE3 duch inizierte Instanzen in E4 Adapter ersetzt werden können. Dies wurde theoretisch wie auch praktisch erklärt und durchgeführt.

Gegenüberstellung E3 und E4

In der folgenden Tabelle werden diverse Themen zu „Dependency Injection“ einander gegenübergestellt. Die Tabelle soll einen Überblick schaffen wo sich E3 und E4 unterscheiden oder wo sie gleich sind.

|  |  |  |
| --- | --- | --- |
| **Thema** | **E3** | **E4** |
| Instanzen holen | Mit statischen Methoden-Aufrufe oder übers Framework (z.B. ViewPart-Hierarchie) Aktives Holen der Instanzen | Injektion der Instanzen über Dependency Injection.  Passiv (Hollywood Prinzip[[8]](#footnote-8)) |
| Kopplung | Enge Kopplung ans Framework | Lose Kopplung ans Framework |
| Context | TODO Marc | Hierarchisch aufgebauter Context (Part Context, Perspective Context, Window Context, Workbench Context, OSGi Context) org.eclipse.e4.core.contexts.IEclipseContext |

Beurteilung Migration

Die Migration ist relativ einfach zu bewerkstelligen, da man in E4 praktisch alles injizieren kann. Die statischen Methodenaufrufe können grösstenteils unkompliziert durch injizierte (@Inject) Instanzen ersetzt werden. Um dies zu erreichen müssen die Parts jedoch vorher auf E4 (POJOs) migriert – also ins Application Model integriert - werden

Adapter

Was wurde erreicht?

Es konnte ein Weg aufgezeigt werden, wie E3 Adapter nach E4 Adapter migriert werden können.

Dies wurde theoretisch wie auch praktisch erklärt und durchgeführt.

Gegenüberstellung E3 und E4

In der folgenden Tabelle werden diverse Themen zu Adapter einander gegenübergestellt. Die Tabelle soll einen Überblick schaffen wo sich E3 und E4 unterscheiden oder wo sie gleich sind.

|  |  |  |
| --- | --- | --- |
| **Thema** | **E3** | **E4** |
| Adapter(manager) Instanz holen | Statischer Methoden-Aufrufe (Platform.getAdapterManager()) um eine Instanz vom IAdapterManager zu erhalten | Injektion der Adapter-Instanz über DI |
| Deklaration der AdapterFactory’s | Im plugin.xml | Im plugin.xml |
| Implementation der AdapterFactory‘s | Klasse die org.eclipse.core.runtime.IAdapterFactory implementiert | Klasse die org.eclipse.core.runtime.IAdapterFactory implementiert |
| Typsicherheit | Cast notwendig | Kein Cast notwendig |

Beurteilung Migration

Die Migration ist – vorausgesetzt die vorherige Migration, also ViewPart nach POJO etc., ist gemacht – ist sehr einfach durchzuführen. Die Aufrufe von Platform.getAdapterManager() können problemlos durch Injizieren von Adapter-Instanzen ersetzt werden.

Es stellt sich nur die Frage, ob die Deklaration und sonstige Handhabung der Adapter-Factories in Zukunft so bleiben wird. Denn gegenüber E3 hat sich nichts verändert, die Factories müssen nach wie vor im plugin.xml deklariert werden.

* + 1. Test

Der Umbau des Parts wurde händisch getestet. Der Part verhielt sich genau gleich wie vor dem Umbau. Die Selektion der Züge wie auch der Betriebspunkte hat nach wie vor funktioniert. Der Umbau wurde als erfolgreich eingestuft.

DI

TODO Screenshots

Adapter

TODO Screenshots

* + 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 anders 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. Risikobeurteilung

TODO

* + 1. Massnahmen

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

Das ursprüngliche Vorhaben maximal 2 Wochen für einen Aspekt in Anspruch zu nehmen wurde über Bord geworfen. Es ist illusorisch die teilweise sehr umfangreichen Themen mit Sicherheit innerhalb von 2 Wochen in der gewünschten, hohen Qualität abzuschliessen. Wir bewerten also Qualität höher als die Quantität. Neu soll ein Aspekt so schnell wie möglich abgeschlossen werden, wenn möglich sollten zumindest die ersten 4 Aspekte abgeschlossen werden können.

* + 1. Fazit

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

Lessons learned

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

Mit diesem Aspekt wurden das Thema „Commands / Handler, Menus, Key Bindings“ behandelt. Es sollte aufgezeigt werden wie diese Punkte in E4 gelöst werden können. Dasselbe sollte für E3 aufgezeigt werden. Mit dem Wissen über die Gegebenheiten in beiden Eclipse Versionen sollte nun ein Vergleich stattfinden und ein Weg aufgezeigt werden wie die Punkte des Aspektes migriert werden können.

* + 1. Definition Abnahmekriterien

Die Abnahmekriterien wurden vom Betreuer folgendermassen definiert:

* Eine beispielhafte Deklaration eines Commands mit E4 Mitteln
* Eine beispielhafte Implementierung eines E4 Handlers
* Eine beispielhafte Deklaration eines Menues mit E4 Mitteln
* Eine beispielhafte Deklaration eines Key Bindings mit E4 Mitteln
* 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.

TODO Fragment.e4xmi ist nicht so praktisch erweiterbar wie das Application Model XMI, man muss Application Model kennen 🡪 Exception

Wenn MenuHandleItem direkt an Window.mainMenu:

java.lang.ClassCastException: org.eclipse.e4.ui.model.application.ui.menu.impl.HandledMenuItemImpl cannot be cast to org.eclipse.e4.ui.model.application.ui.menu.MMenu  
       at org.eclipse.e4.ui.model.application.ui.basic.impl.WindowImpl.eSet(WindowImpl.java:758)  
       at org.eclipse.e4.ui.model.application.ui.basic.impl.TrimmedWindowImpl.eSet(TrimmedWindowImpl.java:121)  
       at org.eclipse.emf.ecore.impl.BasicEObjectImpl.eSet(BasicEObjectImpl.java:1071)  
       at org.eclipse.e4.ui.model.internal.ModelUtils.merge(ModelUtils.java:137)  
       at org.eclipse.e4.ui.model.fragment.impl.StringModelFragmentImpl.merge(StringModelFragmentImpl.java:288)  
       at org.eclipse.e4.ui.internal.workbench.ModelAssembler.processModel(ModelAssembler.java:167)       at org.eclipse.e4.ui.internal.workbench.ResourceHandler.loadMostRecentModel(ResourceHandler.java:220)  
       at org.eclipse.e4.ui.internal.workbench.swt.E4Application.loadApplicationModel(E4Application.java:370)  
       at org.eclipse.e4.ui.internal.workbench.swt.E4Application.createE4Workbench(E4Application.java:220)  
       at org.eclipse.ui.internal.Workbench$5.run(Workbench.java:557)  
       at org.eclipse.core.databinding.observable.Realm.runWithDefault(Realm.java:332)  
       at org.eclipse.ui.internal.Workbench.createAndRunWorkbench(Workbench.java:543)  
       at org.eclipse.ui.PlatformUI.createAndRunWorkbench(PlatformUI.java:149)  
       at ch.sbb.rcsd.client.application.internal.RCSDApplication.start(RCSDApplication.java:33)  
       at org.eclipse.equinox.internal.app.EclipseAppHandle.run(EclipseAppHandle.java:196)  
       at org.eclipse.core.runtime.internal.adaptor.EclipseAppLauncher.runApplication(EclipseAppLauncher.java:110)  
       at org.eclipse.core.runtime.internal.adaptor.EclipseAppLauncher.start(EclipseAppLauncher.java:79)  
       at org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:353)  
       at org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:180)  
       at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
       at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)  
       at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)  
       at java.lang.reflect.Method.invoke(Unknown Source)  
       at org.eclipse.equinox.launcher.Main.invokeFramework(Main.java:629)  
       at org.eclipse.equinox.launcher.Main.basicRun(Main.java:584)  
       at org.eclipse.equinox.launcher.Main.run(Main.java:1438)  
       at org.eclipse.equinox.launcher.Main.main(Main.java:1414)

wenn Command hinzugefügt

java.lang.ClassCastException: org.eclipse.e4.ui.model.application.commands.impl.CommandImpl cannot be cast to org.eclipse.e4.ui.model.application.ui.MUIElement  
       at org.eclipse.e4.ui.model.application.ui.impl.ElementContainerImpl$1.validate(ElementContainerImpl.java:1)  
       at org.eclipse.emf.common.util.BasicEList.addUnique(BasicEList.java:418)  
       at org.eclipse.emf.common.notify.impl.NotifyingListImpl.doAddUnique(NotifyingListImpl.java:325)  
       at org.eclipse.emf.common.notify.impl.NotifyingListImpl.addUnique(NotifyingListImpl.java:309)  
       at org.eclipse.emf.common.util.AbstractEList.add(AbstractEList.java:301)  
       at org.eclipse.e4.ui.model.internal.ModelUtils.mergeList(ModelUtils.java:169)  
       at org.eclipse.e4.ui.model.internal.ModelUtils.merge(ModelUtils.java:126)  
       at org.eclipse.e4.ui.model.fragment.impl.StringModelFragmentImpl.merge(StringModelFragmentImpl.java:288)  
       at org.eclipse.e4.ui.internal.workbench.ModelAssembler.processModel(ModelAssembler.java:167)  
       at org.eclipse.e4.ui.internal.workbench.ResourceHandler.loadMostRecentModel(ResourceHandler.java:220)  
       at org.eclipse.e4.ui.internal.workbench.swt.E4Application.loadApplicationModel(E4Application.java:370)  
       at org.eclipse.e4.ui.internal.workbench.swt.E4Application.createE4Workbench(E4Application.java:220)  
       at org.eclipse.ui.internal.Workbench$5.run(Workbench.java:557)  
       at org.eclipse.core.databinding.observable.Realm.runWithDefault(Realm.java:332)  
       at org.eclipse.ui.internal.Workbench.createAndRunWorkbench(Workbench.java:543)  
       at org.eclipse.ui.PlatformUI.createAndRunWorkbench(PlatformUI.java:149)  
       at ch.sbb.rcsd.client.application.internal.RCSDApplication.start(RCSDApplication.java:33)  
       at org.eclipse.equinox.internal.app.EclipseAppHandle.run(EclipseAppHandle.java:196)  
       at org.eclipse.core.runtime.internal.adaptor.EclipseAppLauncher.runApplication(EclipseAppLauncher.java:110)  
       at org.eclipse.core.runtime.internal.adaptor.EclipseAppLauncher.start(EclipseAppLauncher.java:79)  
       at org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:353)  
       at org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:180)  
       at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
       at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)  
       at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)  
       at java.lang.reflect.Method.invoke(Unknown Source)  
       at org.eclipse.equinox.launcher.Main.invokeFramework(Main.java:629)  
       at org.eclipse.equinox.launcher.Main.basicRun(Main.java:584)  
       at org.eclipse.equinox.launcher.Main.run(Main.java:1438)  
       at org.eclipse.equinox.launcher.Main.main(Main.java:1414)

Seltsames Verhalten bei Fragment und Hinzufügen Menu 🡪 Es steht ein application zwl Window zur Auswahl!?

Nützlich: "org.eclipse.e4.tools.emf.liveeditor“ mit ALT+SHIFT+F9 Application Model

Evtl. Bilder einfügen von Liveeditor

Probleme E4 part.setIconURI("platform:/plugin/ch.sbb.rcsd.client.map.ui/icons/view16/app\_karte.png"); Keine Fehlermeldung wenn falsch

* + 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

Lessons learned

* 1. Aspekt „Eigene Extension Points / Eigene Services“

Mit diesem Aspekt wurden das Thema „Commands / Handler, Menus, Key Bindings“ behandelt. Es sollte aufgezeigt werden wie diese Punkte in E4 gelöst werden können. Dasselbe sollte für E3 aufgezeigt werden. Mit dem Wissen über die Gegebenheiten in beiden Eclipse Versionen sollte nun ein Vergleich stattfinden und ein Weg aufgezeigt werden wie die Punkte des Aspektes migriert werden können.

* + 1. Definition Abnahmekriterien

Die Abnahmekriterien wurden vom Betreuer folgendermassen definiert:

* Eine beispielhafte eigene Deklaration Extension Point mit E4 Mitteln mit einer Nutzung (Extension)
* Eine beispielhafte Bereitstellung und Nutzung eines E4 Services
* 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.

* + 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

Lessons learned

1. Aufräum-Iteration zwischen Aspekt 3 und 4

Nach dem Review vom 08.08.2013 mit dem Betreuer wurde entschieden eine Aufräum-Iteration einzuschieben bevor es mit dem Aspekt 4 weitergeht. Dieser Schritt war notwendig, damit nicht alle Aufräumarbeiten auf den Schluss der Arbeit fallen. Jetzt wo die Themen noch relativ frisch sind, ist es weitaus sinnvoller die Aufräumarbeiten zu starten.

1. Zwischenreview

Am 10.07.2013 wurde mit dem Experten und Betreuer an der Haslerstrasse 30 in Bern ein Zwischenreview durchgeführt. Dabei wurde folgendes präsentiert:

* RCS, als das Rail Control System der SBB
  + Wie sehen die UIs aus
  + Was TODO mehr

1. Aspekt Application Model vs. Advisors

Dieser Aspekt wurde aufgrund von Zeitmangel nicht bearbeitet. Aber teilweise ist das Thema durch den Aspekt 1 abgedeckt.

1. Reflexion
2. Präsentation

# Beurteilungskriterien für Diplompräsentationen

Folgende Kriterien sind massgebend für die Beurteilung der Diplompräsentation. Die Einzelkriterien sind mit jeweils 1 Punkt gewichtet.

### 1. Publikumskontakt

* Der Blickkontakt zum Publikum besteht.
* Die persönliche Präsenz (Mimik/Gestik) ist spührbar.
* Ein geeigneter Standort wurde gewählt.

### 2. Stimme

* Die stimmliche Lautstärke ist der Raumgrösse angepasst
* Die Aussprache ist deutlich (Atmung, Betonung, Sprechpausen)

### 3. Präsentationsmittel

* Die eingesetzten Hilfsmittel sind in Gestaltung und Umfang der Präsentation angepasst

### 4. Aufbau / Gliederung

* Die Gliederung der Präsentation ist ersichtlich (Einleitung, Hauptteil, Abschluss)
* Die Hauptbotschaft und der rote Faden der Präsentation sind erkennbar und verständlich

### 5. Zeitmanagement

* Die zur Verfügung stehende Zeit wird eingehalten

### 6. Gesamtbild

* Der Redner/die Rednerin hinterlässt einen professionellen Eindruck des Auftretens

Verzeichnisse / Quellen

* 1. Abbildungsverzeichnis

[Abbildung 1 E4 Branch Builds 12](#_Toc366071113)

* 1. Quellverzeichnis

Das Quellverzeichnis befindet sich im Projekthandbuch[[9]](#footnote-9)

1. <http://jenkins-ci.org/> [↑](#footnote-ref-1)
2. <http://www.ti.bfh.ch/fileadmin/data/weiterbildung/SWS/Master_Thesis/6_Statusbericht.docx> [↑](#footnote-ref-2)
3. BU = Brawand Ueli, HM =Hoffmann Marc, RM = Rothenbühler Mike [↑](#footnote-ref-3)
4. Vertreten durch Amrhein Beatrice [↑](#footnote-ref-4)
5. <https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.docx> [↑](#footnote-ref-5)
6. <https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.docx> [↑](#footnote-ref-6)
7. <https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.docx> [↑](#footnote-ref-7)
8. <http://en.wikipedia.org/wiki/Hollywood_principle> [↑](#footnote-ref-8)
9. <https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.docx> [↑](#footnote-ref-9)