Berner Fachhochschule
Technik und Informatik

Software-Schule Schweiz

Master Thesis
MT-FS13-05
Migration von Eclipse 3.x nach Eclipse 4

12.09.2013

Projektbericht

Student: Rothenbühler Mike

Betreuer: Hoffmann Marc

Experte: Brawand Ueli

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.

Java, Eclipse RCP, E4, Dependency Injection, Application Model, Migration, RCS

Autor:	Mike Rothenbühler
Version:	1.00
Status:	Freigegeben
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	0.8	MIRO	Aufräumarbeiten
05.09.2013	0.81	MIRO	Aufräumarbeiten
09.09.2013	0.82	MIRO	Aufräumarbeiten
10.09.2013	0.83	MIRO	Aufräumarbeiten, letzte Iteration, Präsentation
11.09.2013	0.90	MIRO	Aufräumarbeiten, Korrekturen, Fazit
12.09.2013	1.00	MIRO	Aufräumarbeiten, Korrekturen, Fazit

MIRO = Mike Rothenbühler

Inhaltsverzeichnis

1. Einleitung

Das Projekt „Migration von Eclipse 3.x nach Eclipse 4“ wird im Rahmen der Master Thesis des „*Master of Advanced Studies in Information Technology*“[1] durchgeführt.

Das Resultat der Master Thesis sind die zwei Dokumente Projektbericht[2] und Projekthandbuch[3].

1.1. 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.

2. Projekt / Projektziele

2.1. 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 3 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 folgende These soll während und nach dem Abschluss der Arbeit überprüft werden:

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] http://www.ti.bfh.ch/de/weiterbildung/sws/mas_it.html

[2] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projektbericht.pdf

[3] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.pdf

2.2. Problemstellung

Da sich mit der Version 4 einiges an Eclipse RCP geändert hat, ist eine Migration nur mit hohem Aufwand 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 muss auch während der Migrationszeit möglich sein. Hierfür müssen Lösungen erarbeitet werden.

Mit den Erfahrungen und Ergebnissen aus der Master Thesis soll eine Migration für grosse Projekte vereinfacht werden.

2.3. Randbedingungen

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

2.4. 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 und weiteren Bahnen verwendet.

2.5. Stakeholder

Rolle	Name	Email
Projektumsetzung	Mike Rothenbühler	michael.rothenbuehler@sbb.ch
Projektbetreuer	Marc Hoffmann	marc.hoffmann@sbb.ch
Experte	Ueli Brawand	ueli.brawand@besonet.ch

Der Projektbetreuer Marc Hoffmann ist seit 2008 im Projekt RCS in verschiedenen Teilprojekten tätig. Er besitzt als Architekt ein umfangreiches Wissen über RCS. Dazu hat er seit 2002 an verschiedenen Eclipse-Projekten gearbeitet.

2.6. Zielvorstellungen

Das Ziel des Projektes ist es, mindestens 4 Aspekte von Eclipse RCP 3 auf Eclipse RCP 4 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 Artefakte sind neben diesem Projektbericht das Projekthandbuch.

Die Aspekte und deren Migration – also der technische Blickwinkel - werden im Projekthandbuch[4] dokumentiert. Die Erfahrungen im Projekt sind in diesem Bericht geschildert.

2.7. Lösungen

Die Lösung eines Aspektes wird jeweils während einer Iteration erarbeitet. Nach Möglichkeit sollen 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).

2.8. Sicherheits- und Datenschutzaspekte

Der Code der SBB ist nicht für Dritte zugänglich. Aus diesem Grund werden nur Ausschnitte aus dem Code publiziert.

[4] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.pdf

3. Risiken

3.1. 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 eng terminiert.	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 der Auftraggeber nicht akzeptieren.
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.
5	Verlust von Stabilität	20 %	Es sind heute keine Probleme betreffend 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 ausfallen.
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 sehr wahrscheinlich gleich bleiben.	4	Das Look and Feel sollte gleich bleiben da Änderungen gegebenenfalls zu Schulungen und Anpassungen von Handbüchern führen.

3.2. 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, wird dieser nicht weiter behandelt. Die Erfahrungen werden dokumentiert und kritisch hinterfragt.
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 müssen 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, falls 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 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.

Die Risiken sollen nach Abschluss von jeder Iteration neu bewertet werden. Falls neue Risiken auftreten, werden auch diese dokumentiert.

4. Teststrategie

Der Erfolg der Bearbeitung aller Aspekte wird mit Tests bewiesen. Die Tests werden mit dem folgenden Testprotokoll dokumentiert:

Bezeichnung	Testfall XY		
Beschreibung	Dieser Test prüft XY		
Voraussetzungen	Applikation ist gestartet		
Fokus	Funktionalität		
Test-Datum	DD.MM.YYYY		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	Applikation starten	Login Dialog erscheint	Login Dialog erschienen
2	Einloggen	Erfolgreiches Einloggen, Hauptfenster mit Infocenter View erscheint	Login erfolgreich, Hauptfenster mit Infocenter View geöffnet
3	…	….	….
Testergebnis	OK		

5. Vorbereitung

5.1. Know-how Aufbau

Das aktuelle zu bearbeitende Projekt RCS ist auf Eclipse 3 aufgebaut. Aufgrund dieser praktischen Erfahrung habe ich ein solides Know-How in diesem Bereich aufgebaut. 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

Autor(en)	**E4 Bücher**	**Ort**	**Verlag**	**Jahr**
Teufel, Marc, Helming, Jonas	Eclipse 4 - Rich Clients mit dem Eclipse 4.2 SDK	Dresden, D	entwickler.press	06.11.2012
Vogel, Lars	Eclipse 4 Application Development	Hamburg, D	Lars Vogel	26.06.2012
	E3 Bücher			
Chris Aniszczyk, Jean-Michel Lemieux, Jeff McAffer	Eclipse Rich Client Platform	Amsterdam, NL	Addison-Wesley Longman	28.06.2010
Stefan Reichert	Eclipse RCP im Unternehmenseinsatz	Heidelberg, D	dpunkt Verlag	17.08.2009

6. Organisatorisches

Folgende Meetings mit dem Experten und dem Betreuer sind seitens Hochschule vorgeschrieben:

- Kickoff
- Zwischenreview (mindestens 1)
- Schlussreview

Mit dem Experten wurde vereinbart, alle 2-4 Wochen einen Statusbericht abzugeben. Dafür wird von der Hochschule ein Template[5] zur Verfügung gestellt. Der Statusbericht wird per E-Mail an den Experten und den Betreuer versandt. Zusätzlich werden die Statusberichte im Repository unter https://github.com/MikeR13/MAS/tree/master/Deliverables abgelegt.

7. Vorgaben

7.1. Beurteilungskriterien

Die in diesem Kapitel aufgeführten Beurteilungskriterien wurden zum einen im Kickoff-Meeting bestimmt und zum anderen von der Hochschule vorgegeben.

Vordergründig wird das methodische Vorgehen bewertet. Die Abschlussarbeit wird gegenüber dem Projekt bzw. der Arbeit für den Arbeitgeber prioritär behandelt. Abschliessend wird die Zielerreichung beurteilt.

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%

[5] http://www.ti.bfh.ch/fileadmin/data/weiterbildung/SWS/Master_Thesis/6_Statusbericht.docx

7.2. Tipps und Vorgaben vom Experten

Der Experte wird während dem Projekt kein Feedback und keine Empfehlungen abgeben. Die Arbeit soll selbständig erarbeitet werden. .

Der Experte erwartet einen Bericht im Umfang von 30 – 50 Seiten mit Informationen zum Verlauf des Projektes. Der Bericht besteht beispielsweise aus folgenden Inhalten:

- Risiken (Erhebung, Beurteilung, …)
- Testing (Art der Tests, erwartete Testresultate, Ergebnisprüfung),
- Methodik/Projektmanagement (Planung, Tracking, Bewertung)
- Lessons learned

In der Arbeit werden nur Referenzen erwähnt, die verwendet wurden. Falls eine Quelle nicht verfügbar ist, soll ein kurzes Abstract erstellt werden.

Die folgenden Tipps stammen aus einem vom Experten vorgängig 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*.“

7.3. Checkliste Dokumente Master Thesis

Von der Berner Fachhochschule wurde die folgende Checkliste abgegeben:

Titelblatt

- SWS-Logo (svg, 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.

8. Projektplan

In der folgenden Tabelle sind sämtliche 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 der Fachhochschule zu Beginn festgelegt wurden. Die weiteren Termine wurden im Laufe des Projektes vereinbart.

Datum	Termin	Beteiligte Personen
02.05.2013	Kickoff Meeting	BU, HM, RM[6]
15.05.2013	Meeting mit Betreuer, Thema: Abstract, Projektantrag	HM, RM
21.05.2013	Abstract	RM
21.05.2013	Projektantrag	RM
07.06.2013	Statusbericht	BU, HM, RM
19.06.2013	Meeting mit Betreuer, Thema: Abnahme Aspekt 1	HM, RM
24.06.2013	Statusbericht	BU, HM, RM
09.07.2013	Meeting mit Betreuer, Thema: Abnahme Aspekt 2	HM, RM
10.07.2013	Meeting mit Betreuer, Thema: Zwischenreview	HM, RM
10.07.2013	Zwischenreview	BU, HM, RM
31.07.2013	Meeting mit Betreuer, Thema: Abnahme Aspekt 3	HM, RM
02.08.2013	Statusbericht	BU, HM, RM
08.08.2013	Meeting mit Betreuer, Thema: Abnahme Aspekt 4, Aufräumarbeiten	HM, RM
16.08.2013	Statusbericht	BU, HM, RM
16.08.2013	Meeting mit Betreuer, Thema: Aufräumarbeiten	HM, RM
29.08.2013	Meeting mit Betreuer, Thema: Aufräumarbeiten	HM, RM
03.09.2013	Meeting mit Betreuer, Thema: Aufräumarbeiten	HM, RM
05.09.2013	Schlussreview	BU, HM, RM
11.09.2013	Meeting mit Betreuer, Thema: Aufräumarbeiten	HM, RM
12.09.2013	Eingabe Präsentationsbedürfnisse	RM
12.09.2013	Abgabe Dokumentation/Anhänge	RM
13.09.2013	Meeting mit Betreuer, Thema: Präsentation	HM, RM
18.09.2013	Testdurchlauf Präsentation vor Team	HM, Team, RM
20.09.2013	Schlusspräsentation	BU (7), HM, RM

Alle Ergebnisse aus den Review-Terminen mit dem Betreuer wurden jeweils als GitHub Issue, welche dem richtigen Milestone[8] zugeordnet sind, erfasst und abgearbeitet.

Die Statusberichte wurden dem Experten jeweils per Mail gesendet. Alle Statusberichte sind dazu im Verzeichnis https://github.com/MikeR13/MAS/tree/master/Deliverables abgelegt.

[6] BU = Brawand Ueli, HM =Hoffmann Marc, RM = Rothenbühler Mike

[7] Vertreten durch Amrhein Beatrice

[8] https://github.com/MikeR13/MAS/issues/milestones

Aufgrund der Termine, der geplanten Abwesenheiten 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 werden die nötigen Vorbereitungen getroffen. Es wird definiert, welche Aspekte im Rahmen der Arbeit behandelt werden sollen. In den darauffolgenden 12 Wochen sollen die einzelnen Aspekte fortlaufend bearbeitet werden. Die letzten 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.

Folgende Iterationen sind geplant:

- Vorbereitung
- Aspekt 1 – x (nach Möglichkeit 5, jedoch mindestens 4) bearbeiten
- Abschlussarbeiten

Die geleisteten Arbeiten werden jeweils in Stunden pro Tag im Dokument https://github.com/MikeR13/MAS/blob/master/Planung/Geleistete%20Arbeiten.xlsx erfasst. Im Dokument https://github.com/MikeR13/MAS/blob/master/Planung/Planung.xlsx werden pro Woche die tatsächlich geleisteten Stunden den geplanten Stunden gegenübergestellt. So kann geprüft werden, ob die geplanten Stunden geleistet wurden. Falls eine Verzögerung im Plan auftritt, müssen Massnahmen eingeleitet werden, damit die Arbeiten zeitgerecht fertiggestellt werden können.

Die folgende Tabelle wurde der Planung.xlsx-Datei (aus der Projekt-Planung) entnommen. Die Tabelle zeigt auf, dass während dem Aspekt „Commands / Handler, Menus, Key Bindings“ die geplante Zeit nicht eingehalten werden konnte. Als Massnahme wurde die wöchentliche Arbeitszeit auf 2 Tage erhöht. Dazu habe ich meine Ferien um 2 Tage gekürzt und teilweise auch in den Ferien gearbeitet

Datum Start	Geplantes Vorhaben	geplanter Aufwand in h	tatsächlicher Aufwand in h	Differenz
22.04.2013	Planung ausarbeiten, Aufgabe konkretisieren, Kickoff Meeting vorbereiten	18.0	16.0	-2.0
29.04.2013	Kickoff, Workspace aufsetzen	18.0	11.0	-7.0
06.05.2013	Abstract, Projektantrag, Planung	18.0	25.0	7.0
13.05.2013	Abstract, Projektantrag, Planung	18.0	13.0	-5.0
20.05.2013	Aspektermittlung, Projekt-Setup	18.0	22.0	4.0
27.05.2013	Aspektermittlung, Projekt-Setup	18.0	9.0	-9.0
03.06.2013	Iteration 1, Aspekt 1 angehen	18.0	19.0	1.0
10.06.2013	Iteration 1, Aspekt 1 angehen	18.0	16.0	-2.0
17.06.2013	Iteration 2, Aspekt 2 angehen	9.0	16.0	7.0
24.06.2013	Iteration 2, Aspekt 2 angehen	27.0	16.5	-10.5
01.07.2013	Iteration 3, Aspekt 3 angehen	18.0	17.0	-1.0
08.07.2013	Iteration 3, Aspekt 3 angehen	18.0	16.0	-2.0
15.07.2013	Iteration 4, Aspekt 4 angehen	9.0	6.5	-2.5
22.07.2013	Iteration 4, Aspekt 4 angehen	27.0	12.0	-15.0
29.07.2013	Iteration 5, Aspekt 5 angehen	18.0	21.5	3.5
05.08.2013	Iteration 5, Aspekt 5 angehen	18.0	21.0	3.0
12.08.2013	Abschlussarbeiten	18.0	27.0	9.0
19.08.2013	Ferien	0.0	3.0	3.0
26.08.2013	Ferien	0.0	24.0	24.0
02.09.2013	Abschlussarbeiten	18.0	38.5	20.5
09.09.2013	Abschlussarbeiten	18.0	32.0	14.0
12.09.2013	**ABGABE Dokumentation/Anhänge**			
16.09.2013	Präsentation	18.0	Unbekannt	-18
Total		378.0	382.0	4.0

Der Tabelle ist zu entnehmen, dass gegen Ende der Arbeit wesentlich mehr Zeit als geplant investiert wurde. Das war auch nötig, um die gewünschte Qualität der Arbeit zu erreichen. Die aufgewendete Zeit der letzten Woche, konnten leider nicht mehr erfasst werden, da die Abgabe der Dokumente bereits am 12.09.2013 stattfindet.

9. Setup

Es wurde bestimmt die Dokumentation der Arbeit (Projektbericht und Projekthandbuch[9]) mittels Microsoft Office Tools zu erstellen. Der Sourcecode wird in Java Files gehalten.

Um die erarbeiteten Dokumente und weitere 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 zusätzliches Feature: die Erfassung von Milestones und Issues. Diese Funktionalität wird in diesem Projekt zum Einsatz kommen. Pro Iteration wird ein Milestone mit bestimmtem Enddatum erfasst. Zu jedem Milestone werden die benötigten Issues - oder in diesem Kontext besser Tasks - erfasst. Nach Erledigung eines Tasks, wird dieser geschlossen, er erhält den Status closed. Die Issues werden sequentiell abgearbeitet.

[9] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.pdf

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 aktuell gehalten. Wird der automatische Merge aufgrund von Konflikten nicht erfolgreich durchgeführt, wird manuell Einfluss genommen. Bei einem missglückten automatischen Merge werden die beteiligten Personen per E-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. Die Tests werden mindestens einmal täglich oder direkt nach einem Commit auf dem Branch ausgelöst. Der Verursacher des Fehlers – bzw. der Commiter – wird per E-Mail über Fehler beim Test benachrichtigt.

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

Die URL[10] 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[11]-Buildserver. Die Builds können unter der folgenden URL gestartet, verwaltet und gelesen 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 manuell zu starten:

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

In der folgenden Abbildung wird die Übersicht über die letzten Builds auf dem E4 Branch dargestellt:

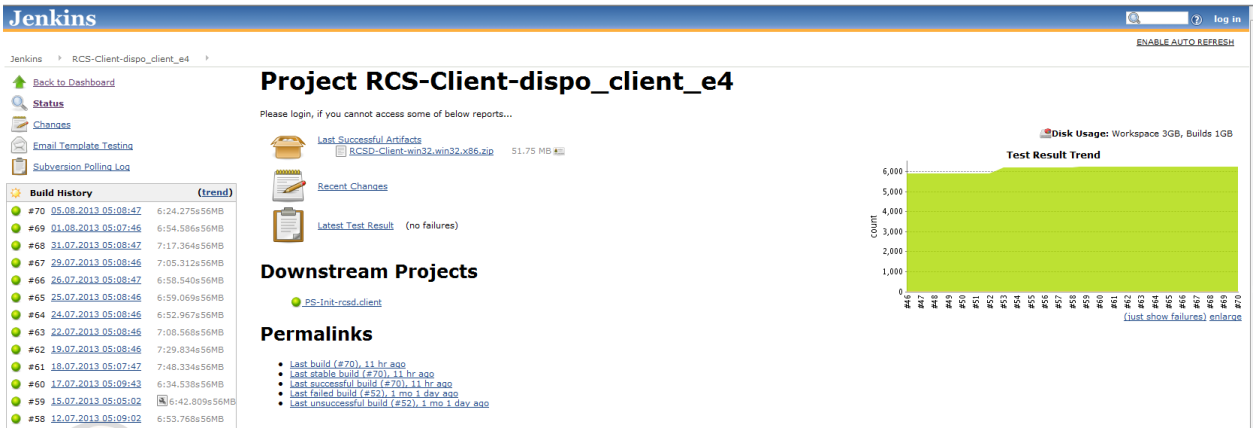

Abbildung 1 E4 Branch Builds

[10] Die URLs sind ausschliesslich aus dem Netz der SBB erreichbar

[11] http://jenkins-ci.org/

10. Aspekt Ermittlung

Bei der Ermittlung der Aspekte hat der Autor vorgängig selbständig gearbeitet. Nachfolgend wurden während drei Sessions mit dem Betreuer weitere Aspekte bearbeitet und diskutiert.

Das Brainstorming ergab folgende möglichen Aspekte:

- 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 weitere Notizen sind unter
https://github.com/MikeR13/MAS/tree/master/Unterlagen_Notizen/Brainstrorming_Aspekte
abgelegt.

Folgende Aspekte wurden bestimmt (Da der Betreuer zugleich den Auftraggeber vertritt, wurden die zu behandelten Aspekte von ihm 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 wie in E3 Instanzen von Services oder sonstigen Framework-Objekten geholt werden. Dies wird dann mit den E4 Möglichkeiten verglichen.
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? Dazu sollen die Services in E3 und E4 verglichen werden. Wie können diese migriert werden?
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 bearbeitet wird, wird dieser nicht mehr verändert.

11. 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?
- Fazit

Detaillierte Information über die Migrationsmöglichkeiten und die Beantwortung von einzelnen Fragen sind jeweils im Projekthandbuch[12] zu finden.

[12] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.pdf

11.1. Aspekt „Mixing E3/E4“

In dieser Iteration wurden die diversen Möglichkeiten E3 und E4 Komponenten in derselben Applikation zu mischen untersucht. Das Mixing ist für grosse Anwendungen, wie bspw. RCS, eine zwingende Voraussetzung.

11.1.1. Definition Abnahmekriterien

Die Abnahmekriterien wurden vom Betreuer folgendermassen definiert:

1. Möglichkeiten aufgeführt und kritisch bewertet und beschrieben
2. Prototyp und Demo

11.1.2. Dauer der Iteration

Die Iteration war in der Periode

03.06.2013 – 16.06.2013

geplant, tatsächlich gedauert hat sie

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 Kapitel 11.1.5 aufgeführt.

11.1.3. Resultate

In diesem Kapitel werden kurz die Resultate aus der Aspekt-Bearbeitung aufgezeigt. Die Resultate und der Prozess der Ergebnisse, sowie die ausführliche Dokumentation zu diesem Aspekt sind im Projekthandbuch[13] 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 erklärt, wie auch praktisch durchgeführt. Die für interessant erklärten Wege wurden in RCS ausimplementiert.

Gegenüberstellung E3 und E4

In der folgenden Tabelle werden diverse Themen zu „Mixing E3/E4“ einander gegenübergestellt. Die Tabelle schafft einen Überblick 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 Fragment Editor
Modeled UI	Nein	Ja

[13] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.pdf

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)

Beurteilung

Dieser Aspekt hat gezeigt, dass es durchaus möglich ist E3 und E4 zu mischen. Diese Erkenntnis ist äusserst wichtig, denn wäre dies nicht möglich wäre eine Migration einer grösseren Applikation – wie zum Beispiel RCS – schlichtweg nicht möglich. Der Bearbeitung der nächsten Aspekte steht also nichts im Weg.

11.1.4. Test

Um die Beibehaltung der Funktionalität zu prüfen wurden Szenarien von Tests ausgewählt die der typische Anwender von RCS sicherlich mehrmals täglich durchläuft. Im Folgenden werden die durchgeführten Testfallszenarien dokumentiert.

Bezeichnung	Testfall A1_1		
Beschreibung	Login und ein ZWL Fenster in RCS öffnen, prüfen, ob sich alles noch wie vorher verhält,		
Voraussetzungen	Entsprechendes ZWL wurde bereits einmal geöffnet		
Fokus	Funktionalität		
Test-Datum	18.06.2013		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	Applikation starten	Login Dialog erscheint	Login Dialog erschienen
2	Einloggen	Erfolgreiches Einloggen, Hauptfenster mit Infocenter View erscheint	Login erfolgreich, Hauptfenster mit Infocenter View geöffnet
3	ZWL BE-NBS* zum Öffnen auswählen	ZWL öffnet sich Alle Züge werden angezeigt	ZWL geöffnet Die Züge werden angezeigt
4	ZWL Verhalten prüfen	Die Ist-Linie verschiebt sich mit der Zeit nach unten Die Züge bewegen sich	Ist-Linie verschiebt sich Die Züge bewegen sich
5	Zug selektieren und mittels CTRL+7 Shortcut Zugfahrteditor öffnen	Farbe der Zuglinie ändert sich Zugfahrteditor wird geöffnet Der richtige Zug wird im Editor angezeigt	Farbe der Zuglinie geändert Zugfahrteditor geöffnet Richtiger Zug wird im Editor angezeigt
Testergebnis	OK		

Bezeichnung	Testfall A1_2		
Beschreibung	Bahnhofinfo öffnen und Funktionalität prüfen		
Voraussetzungen	Applikation ist gestartet, entsprechend berechtigter Benutzer ist eingeloggt		
Fokus	Funktionalität		
Test-Datum	18.06.2013		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	Bahnhofinfo öffnen mittels CTRL+5	Bahnhofinfo Dialog öffnet sich Bahnhofinfo ist selektiert Auswahl Betriebspunkte möglich	Bahnhofinfo Dialog geöffnet Bahnhofinfo selektiert Auswahl BPs möglich
2	Eingabe BN	Die Möglichkeiten zur Auswahl beschränken sich auf Bern	Nur noch Bern zur Auswahl
3	Enter Taste drücken	Bahnhofinfo Bern öffnet sich Bahnhof Bern wird angezeigt	Bahnhofinfo Bern geöffnet Bahnhof Bern wird angezeigt
4	Gleis auswählen und Tooltip prüfen	Tooltip erscheint	Tooltip erscheint
5	Kontextmenu Eigen-schaften aufrufen	Eigenschaften Dialog wird geöffnet	Eigenschaften Dialog geöffnet
Testergebnis	OK		

Bezeichnung	Testfall A1_3		
Beschreibung	Langzeitlauftest		
Voraussetzungen	Applikation ist gestartet, entsprechend berechtigter Benutzer ist eingeloggt, 1 ZWL geöffnet, 1 HGBP geöffnet, Umgebung wird nicht täglich deployt.		
Fokus	Stabilität		
Test-Datum	18.06.2013 - 19.06.2013		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	24 h laufen lassen	Applikation läuft noch ZWL Ist-Linie am richtigen Ort HGBP Ist-Linie ist am richtigen Ort	Applikation läuft noch ZWL Ist-Linie am richtigen Ort HGBP Ist-Linie ist am richtigen Ort
Testergebnis	OK		

Bezeichnung	Testfall A1_4		
Beschreibung	Performance „Gespeicherte Fensteranordnung“ öffnen		
Voraussetzungen	Applikation ist gestartet, entsprechend berechtigter Benutzer ist eingeloggt, die folgende , Fensteranordnung wurde definiert: **ZWL**: Ablis ½ ZUBZ, ZZ-Test_HOT_ZWL, BS-Adler,BN – NBS-AA(KLW), **SSP**: Gurten_MITTE_BLZLZ, **HGBP**: Bern FstZ BN, **SAR:** Bern Fst		
Fokus	Performance		
Test-Datum	09.09.2013		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	E3 Fensteranordnung öffnen, Zeit messen, dieser Schritt soll 5	Zeiten E3	3.7 Sekunden 4.0 Sekunden 3.5 Sekunden 3.9 Sekunden

	Mal durchgeführt werden		4.5 Sekunden
2	E4 Fensteranordnung öffnen, Zeit messen, dieser Schritt soll 5 Mal durchgeführt werden	Zeiten E4	11.09 Sekunden 11.24 Sekunden 12.13 Sekunden 15.27 Sekunden 16.03 Sekunden
Testergebnis	Nicht OK		

Bezeichnung	Testfall A1_5		
Beschreibung	Heap „Gespeicherte Fensteranordnung“ öffnen		
Voraussetzungen	Applikation ist gestartet, entsprechend berechtigter Benutzer ist eingeloggt, die folgende , Fensteranordnung wurde definiert: **ZWL**: Ablis ½ ZUBZ, ZZ-Test_HOT_ZWL, BS-Adler,BN – NBS-AA(KLW), **SSP**: Gurten_MITTE_BLZLZ, **HGBP**: Bern FstZ BN, **SAR:** Bern Fst		
Fokus	Performance		
Test-Datum	09.09.2013		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	E3 Fensteranordnung öffnen, E4 Fensteranordnung öffnen Heapverbrauch vergleichen	Ähnlich viel Heap benötigt	E3: 162.0 MB E4: 337.2 MB
Testergebnis	Nicht OK		

Die Testfälle A1_4 und A1_5 zeigen, dass die Performance von E4 Anwendungen nicht annähernd der Performance von E3 entspricht.
Der Test A1_4 zeigt, dass das Öffnen der 7 Fenster auf E4 dreimal solange dauert wie auf E3. Bei einer etwaigen Migration von RCS muss mit den Anwendern besprochen werden, ob das akzeptierbar ist. Die E4 Tests wurden auf der Plattform 4.3 durchgeführt, dies sei die wesentlich performantere Version als 4.2.
Der Test A1_5 zeigt, dass Eclipse 4 – bei einem identischen Szenario – mehr als doppelt so viel Speicher verbraucht wie E3. Hier muss geklärt werden, wieso dies der Fall ist.

11.1.5. Probleme

In diesem Kapitel wird beschrieben welche Probleme aufgetreten sind, die das Voranschreiten der Arbeiten behindert hat. Es werden 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. Das Evaluieren einer Seite für den Download hat Zeit beansprucht. 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 hilfreich.

Veraltete, falsche und ungenaue Beschreibungen

Viele Quellen im Internet erwiesen sich als veraltet und teilweise auch als 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.platform* gefunden.
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 hinterfragen, 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 dem Application Model ein Window über ein Fragment hinzufügen kann. Da bedeutete teilweise, dass Neuland betreten und Versuche gestartet werden mussten.

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* versucht und es gab keine Exception mehr. Dasselbe hat mit dem Ansatz „LegacyIDE.e4xmi mit Prozessor“ ohne Probleme funktioniert.

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.

Unterschätzung Aufwand Aspekt

Ich habe ganz klar unterschätzt, wie aufwändig die Bearbeitung dieses Aspektes sein wird.

11.1.6. Erfahrungen

In diesem Kapitel werden Ereignisse erwähnt die stattgefunden haben, aber nicht massgebend als Probleme angeschaut werden.

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 vorhanden sind. Beim Hauptfenster befindet sich zum Beispiel neu der Menüpunkt „Ansicht“ in der Toolbar. Die Stellungnahme hierzu von Eclipse: *Dies sei 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.

11.1.7. Risikobeurteilung

In diesem Kapitel werden die Risiken anhand der Erkenntnisse in dieser Iteration neu beurteilt

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 nicht eingehalten, er wurde um 3 Tage überschritten. Dies wurde jedoch toleriert und wird in den nächsten Aspekten wieder eingehalten. Hier sind also keine Massnahmen notwendig.
3	Funktionalität geht verloren	Mit den bisherigen Migrationsschritten konnte kein Verlust an Funktionalität festgestellt werden.
4	Performanceeinbussen	Die Performance hat ganz klar gelitten. E4 ist wesentlich langsamer als E3. Hier muss die Ursache gefunden und gefixt werden. Falls dies nicht möglich ist, so muss mit den Anwendern geklärt werden, ob dies tolerierbar ist. Dies wird ausserhalb dieser Masterarbeit geklärt.
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.

11.1.8. Massnahmen

Da die Überschreitung der Iterationszeit nicht als fatal einzustufen ist und die Ziele erreicht sind, wurden keine Massnahmen definiert. Aus Risikosicht hat sich nichts geändert.

11.1.9. Lessons learned

Diese Iteration hat wieder einmal gezeigt, dass bei der Planung zwingend eine Reserve einberechnet werden muss. Die Arbeiten können immer wieder durch ungeplante Ereignisse verzögert werden.

11.1.10. Fazit

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

Aspekt 2 kann mit gutem Gewissen in Angriff genommen werden.

11.2. Aspekt „Adapter / Dependency Injection“

Mit diesem Aspekt wurde 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 welche Möglichkeiten dieser Mechanismus bietet und wie die bestehenden statischen Aufrufe zum Holen der Services und anderen Framework-Komponenten durch Dependency Injection ersetzt werden.

11.2.1. 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
- Beschreibung des Aspektes im Handbuch
- Aktualisierter Projektbericht

11.2.2. Dauer der Iteration

Die Iteration war in der Periode

17.06.2013 – 30.06.2013

geplant, tatsächlich gedauert hat 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 (7 Tage gegenüber ursprünglicher Planung) 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

Weitere Erklärungen zu diesen Punkten sind im Kapitel Probleme aufgeführt.

11.2.3. Resultate

In diesem Kapitel werden kurz die Resultate aus der Aspekt-Bearbeitung aufgezeigt. Die Resultate und der Weg zu diesen Resultaten sowie die ausführliche Dokumentation zu diesem Aspekt sind im Projekthandbuch[14] 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 in E3 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 schafft einen Überblick:

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 Prinzip15)
Kopplung	Enge Kopplung ans Framework	Lose Kopplung ans Framework
Context	Bei E3 existiert der Context so nicht (hat nichts mit dem Bindig Context zu tun)	Hierarchisch aufgebauter Context (Part Context, Perspective Context, Window Context, Workbench Context, OSGi Context) org.eclipse.e4.core.contexts.IEclipseContext Kann auch als Service Broker betitelt werden.

Beurteilung Migration

Die Migration ist relativ einfach zu bewerkstelligen, da in E4 praktisch alles injiziert werden 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 erklärt wie auch praktisch durchgeführt.

[14] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.pdf

[15] http://en.wikipedia.org/wiki/Hollywood_principle

Gegenüberstellung E3 und E4

In der folgenden Tabelle werden diverse Themen zu Adapter einander gegenübergestellt.

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 vollzogen – 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.

11.2.4. Test

Im Folgenden werden die durchgeführten Testfallszenarien dokumentiert.

Bezeichnung	Testfall A2_1		
Beschreibung	Öffnen der geografischen Karte ohne vorherige Selektion eines Zuges		
Voraussetzungen	Applikation ist gestartet, entsprechend berechtigter Benutzer ist eingeloggt		
Begründung für Test	Mit der Migration wurden die Adapter ausgewechselt und die aktuelle Selektion wird neu durch Dependency Injection gemeldet. Die Selektion eines Zuges und die entsprechende Adaption desselben könnte neu schief laufen		
Fokus	Funktionalität		
Test-Datum	07.07.2013		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	Geografische Karte öffnen über Menu Fenster – Geografische Karte öffnen	Karte öffnet sich	Karte geöffnet
2	Ein ZWL öffnen und einen Zug selektieren	Verlauf des Zuges wird in geografischer Karte angezeigt	Verlauf des Zuges in geografischer Karte angezeigt
Testergebnis	OK		

Die folgende Abbildung zeigt auf, dass bei der Selektion eines Zuges im ZWL (rechts) die Zugfahrt auf der geografischen Karte angezeigt wird.

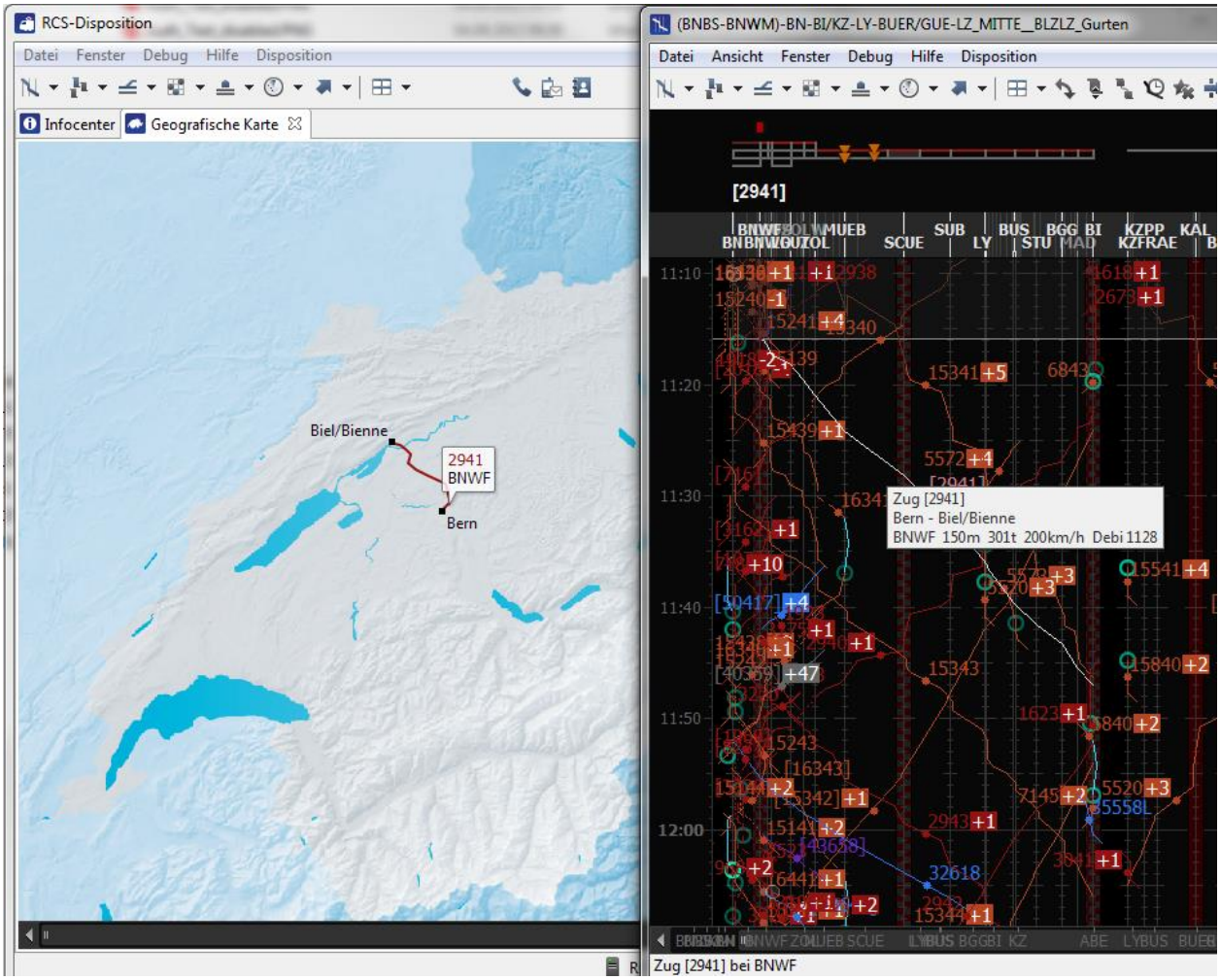

Abbildung 2 Testfall A2_1 Screenshot

11.2.5. 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 ein Wechsel auf 4.3 sinnvoll ist 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/ 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 Entscheid des Versionssprungs war schlichtweg auf dem aktuellen Stand zu sein

Dieser Entschluss hat zu ungeahnten Aufwänden bei der Migration von 4.2 zu 4.3 geführt.
Beim ersten Versuch auf 4.3 zu wechseln bestand 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.

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

Krankheit

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

Umfang Thema

Auch das Thema Dependency Injection bietet viel mehr als ursprünglich angenommen.

11.2.6. Erfahrungen

In diesem Kapitel werden Ereignisse erwähnt die stattgefunden haben, aber nicht unbedingt als Probleme angeschaut werden.

Nicht alles funktioniert einwandfrei mit Targetplattform 4.3

Wenn man im Eclipse Juno, also 4.2, versucht die Targetplattform auf 4.3 zu wechseln, erzeugte 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 führt. 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.

11.2.7. Risikobeurteilung

In diesem Kapitel werden die Risiken anhand der Erkenntnisse in dieser Iteration neu beurteilt

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 4 Tage überschritten. Die Verzögerungen sind erklärbar. Es könnte sein, dass Aspekt 5 geopfert werden muss, um die Arbeit fertig stellen zu können. Mit diesem Vorrat an Zeit wird die Arbeit deshalb rechtzeitig fertig gestellt werden können.
3	Funktionalität geht verloren	Mit den Migrationsschritten der Adapter und Injektion der Instanzen konnte kein Verlust an Funktionalität festgestellt werden.
4	Performanceeinbussen	Gegenüber Aspekt 1 hat sich hier nicht geändert, da nur eine bestimmte View migriert wurde. Hier ist die Performance nicht messbar
5	Verlust von Stabilität	Gegenüber Aspekt 1 hat sich hier nichts geändert, da nur eine bestimmte View migriert wurde. Die wirkt sich nicht auf die Stabilität der Applikation aus.
6	Verlust von Usability	Die E3 View und die E4 verhalten sich genau gleich
7	Wartbarkeit nimmt ab	Der Code ist wartbarer geworden, da zum einen ziemlich viel Code eingespart werden kann (z.B. Adapter).
8	Testbarkeit nimmt ab	Der Code ist ganz klar testbarer geworden, die Instanzen von Objekten können jetzt in JUnit-Tests einfach durch Mocks ersetzt werden. Mit den statischen Methodenaufrufen war das vorher nicht so einfach möglich.
9	Look and Feel wird von Anwendern nicht toleriert	Die E3 geografische Karte und die E4 geografische Karte sind identisch.

Weitere, neue Risiken konnten nicht festgestellt werden.

11.2.8. Massnahmen

Die geplante Iterationszeit wurde wiederum überschritten. Es dürfen deshalb keine zusätzlichen Arbeiten, als die geplanten ausgeführt werden. Auf weitere Versionssprünge muss aufgrund dessen verzichtet werden.

11.2.1. Lessons learned

Die wichtigste Lektion in dieser Iteration ist ganz klar: keine ungeplanten Arbeiten in eine Iteration einbinden. Mit der ungeplanten Arbeit ist die Migration der Targetplattfom von 4.2 auf 4.3 gemeint. Es war eigentlich völlig klar, dass mich diese Aktion auch ohne Probleme ein paar Stunden kosten würde. Diese Stunden habe ich einfach vorher nicht eingeplant. Deshalb noch einmal: KEINE ungeplanten Arbeiten mehr!

11.2.2. Fazit

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

11.3. 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.

11.3.1. 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

11.3.2. Dauer der Iteration

Die Iteration war in der Periode

01.07.2013 – 14.07.2013

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 10 Tage überschritten! Diese Überschreitung ist eindeutig zu hoch, es ist auch bereits die dritte (!) Zeitüberschreitung einer Iteration. Dies ist absolut nicht tolerierbar und muss wiederum begründet werden. Dazu müssen Massnahmen getroffen werden, um eine weitere Zeitüberschreitung zu verhindern. Diese Überschreitung ist erklärbar durch:

- Eingeplante Zeit konnte nicht eingehalten werden
- Fokus verloren, diverse Context-Switches
- E3 Know-How in diesem Bereich überschätzt
- Nicht eingeplanter Zwischenreview Termin
- Hitze

11.3.3. Resultate

Was wurde erreicht?

In dieser Iteration konnte aufgezeigt werden wie Commands, Handlers und Menus von E3 nach E4 migriert werden können. Das Thema wurde theoretisch, wie auch praktisch bearbeitet.

Gegenüberstellung E3 und E4

In der folgenden Tabelle werden diverse Themen zu Commands / Handler, Menus, Key Bindings einander gegenübergestellt.

Thema	E3	E4
Deklaration der Items (Command, Menu, etc.)	Im plugin.xml	Im Application Model
Command	Extension Point org.eclipse.ui.commands im plugin.xml	Im Application Model direkt unter Application
Handler	Extension Point org.eclipse.ui.handlers im plugin.xml	Im Application Model direkt unter Application, innerhalb von Part Descriptors, innerhalb eines Windows oder innerhalb eines Parts
Implementation Handlerklasse	Handlerklasse muss *org.eclipse.core.commands.IHandler* implementieren, Vielfach wird *org.eclipse.core.commands.AbstractHandler* erweitert.	POJO, Auszuführenden Methode mit *@Execute* annotiert, optional eine Methode mit *@CanExecute* annotieren
Referenzierung Handlerklasse	Über Extension Point org.eclipse.ui.handlers im plugin.xml	Über Bundleclass-Notation aus Handler, DirectMenuItem oder DirectToolItem im Application Model
Menu	Extension Point org.eclipse.ui.menus im plugin.xml, mittels locationURI wird bestimmt um was für ein Menu es sich handelt (Menu, Popup, Toolbar) und wo es hin soll (Main-Menu, Main-Toolbar, etc.)	Im Application Model innerhalb eines Windows, als Menu innerhalb eines Parts, oder als PopUp Menu innerhalb eines Parts
Toolbar	Siehe Menu, in der locationURI wird einfach toolbar angegeben	Im Application Model innerhalb von TrimmedWindow's (TrimBars) oder innerhalb von Parts (Option Toolbar muss ausgewählt werden)
Binding	Extension Point org.eclipse.ui.bindings im plugin.xml	Im Application Model innerhalb Application (BindingTables)
Binding Context	Extension Point org.eclipse.ui.contexts im plugin.xml	Im Application Model innerhalb Application (Binding Context)
Action	Nicht mehr brauchen!!	Es gibt nichts dergleichen.

Beurteilung Migration

Die Migration ist nach anfänglichen Schwierigkeiten gelungen. Mir war zuerst nicht klar, welche Items im Fragment erstellt werden müssen. Die Evaluation hat viel Zeit in Anspruch genommen. Die Migration selbst, war schlussendlich leicht zu bewältigen.

Es stellt sich aber für grosse Anwendungen, wie RCS die Frage, ob der Aufwand der manuellen Migration vertretbar ist. Sehr wahrscheinlich ist das nicht der Fall. Man müsste hier Tools verwenden, die uns sämtliche Menu, Command etc. Einträge im plugin.xml nach Einträgen ins fragment.e4xmi übersetzen. Ein Thema, dass hierzu in Betracht gezogen werden muss, ist XLS Transformation.

11.3.4. Test

Im Folgenden werden die durchgeführten Testfallszenarien dokumentiert.

Bezeichnung	Testfall A3_1		
Beschreibung	Öffnen der geografischen Karte über Menupunkt		
Voraussetzungen	Applikation ist gestartet, entsprechend berechtigter Benutzer ist eingeloggt		
Begründung für Test	Mit der Migration wurden das Menu, das Command und der Handler ausgewechselt. Es wurde eine Handlerklasse erstellt, die MapView Referenz auf den DIViewPart entfernt und zum POJO gemacht. All diese Änderungen können dazu führen, dass der Menupunkt nicht mehr erscheint oder nicht ausführbar ist.		
Fokus	Funktionalität		
Test-Datum	30.07.2013		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	In Hauptfenster Menu Fenster anklicken öffnen	Menu Geografische Karte erscheint	Menu Geografische Karte erscheint
2	Menu Geografische Karte drücken	Geografische Karte öffnet sich	Geografische Karte geöffnet
Testergebnis	OK		

Bezeichnung	Testfall A3_2		
Beschreibung	Öffnen der geografischen Karte über Shortcut		
Voraussetzungen	Applikation ist gestartet, entsprechend berechtigter Benutzer ist eingeloggt		
Begründung für Test	Zusätzlich zu Testfall A3_1: Es wurde ein Key Binding zum Öffnen der Karte erstellt, die muss funktionieren		
Fokus	Funktionalität		
Test-Datum	11.0.9.2013		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	In Hauptfenster den Shortcut Ctrl+F12 betätigen	Geografische Karte öffnet sich	Geografische Karte geöffnet
Testergebnis	OK		

Die folgende Abbildung zeigt wie der Menupunkt Geografische Karte zur Verfügung steht. Der E3 Menupunkt wurde bewusst da gelassen um zu aufzuzeigen, dass es wirklich der E4 Menupunkt ist.

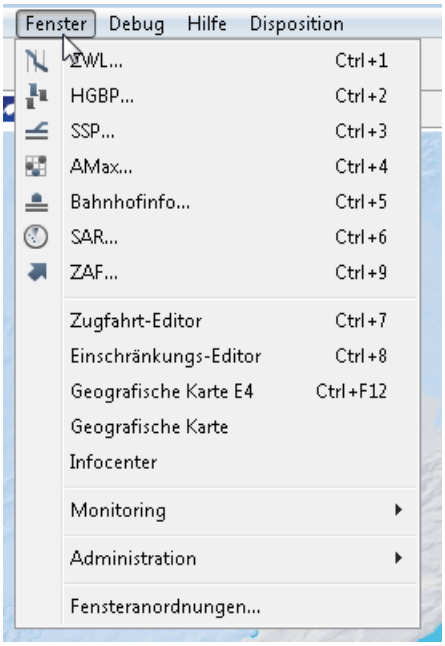

Abbildung 3 Testfall A3_1 und A3_2 Menupunkt Screenshot

Diese Abbildung zeigt, dass die Geografische Karte nach Betätigen des Menupunktes auch angezeigt wird:

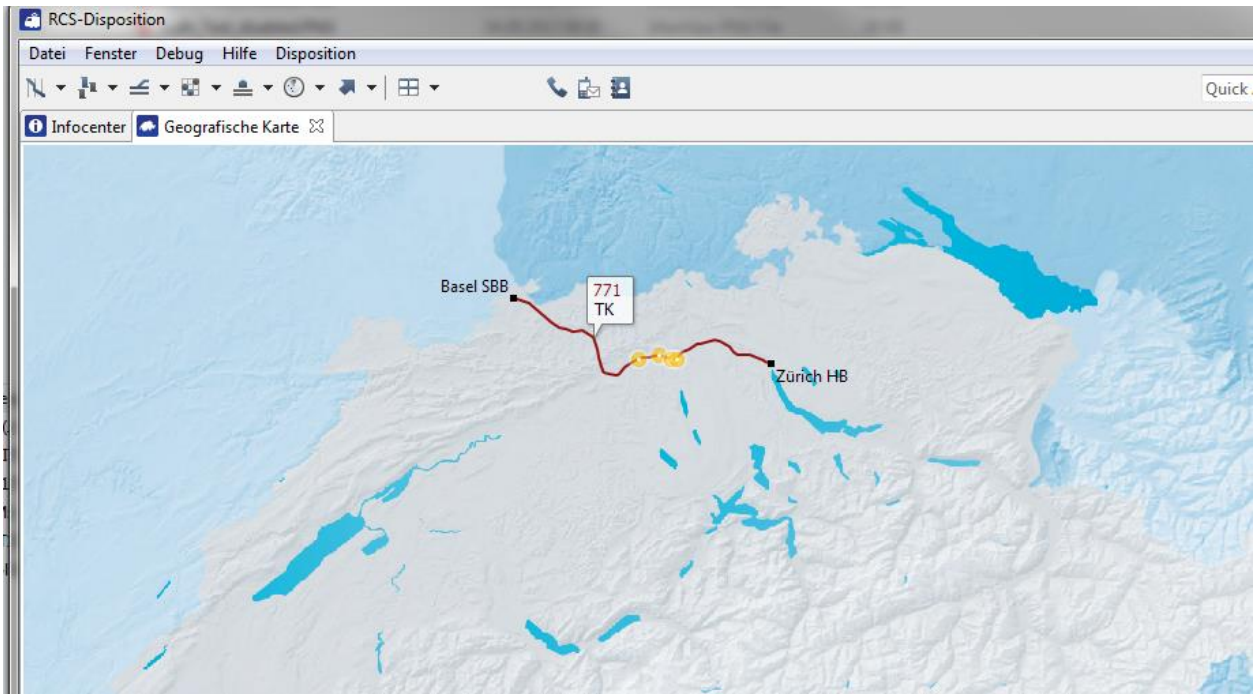

Abbildung 4 Testfall A3_1 Geografische Karte Screenshot

11.3.5. Probleme

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

Eingeplante Zeit konnte nicht eingehalten werden

Die von mir ursprünglich eingeplante Arbeitszeit war zu optimistisch. In jeder Woche dieser Iteration konnte ich die geplanten Zeiten nicht erreichen. Dies ist zum einen auf erhöhte Präsenzzeiten in der Arbeit, aber auch auf private Termine zurückzuführen.

Fokus verloren, diverse Context-Switches

Aus unerklärlichen Gründen habe ich in dieser Iteration vielfach von Issue zu Issue hin- und hergesprungen.

E3 Know-How in diesem Bereich überschätzt

Ich habe eindeutig unterschätzt, wie vielfältig dieser Aspekt ist. Mir waren nicht alle Konzepte – vor allem im E3 Bereich - bekannt und deshalb war eine ungeplante Einarbeitung von mehreren Stunden nötig.

Nicht eingeplanter Zwischenreview Termin

Am Anfang dieser Iteration fand das Zwischenreview mit dem Experten statt. Den Aufwand für die Vorbereitung und die Durchführung, wie auch die Nachbearbeitung habe ich schlichtweg nicht eingeplant.

Hitze

Die enorme Hitze in dieser Zeit hat merkbar meiner Produktivität geschadet.

Fragment Editor

Die Erweiterung des Application Models über Einträge in der Fragment-Datei (hier *fragment.e4xmi*) ist nicht so einfach wie in der Application Model-Datei (hier *Legacy.e4xmi*). Hier ist das Tooling noch nicht soweit und bietet nur beschränkt Unterstützung. Um einen Eintrag zu verfassen, ist eine gute Kenntnis des Application Model notwendig.
Bspw. Beim Erfassen eines *MenuHandleItem* direkt unterhalb von *Window.mainMenu* , wird dies erst zur Runtime bemerkt, die Fehlermeldung ist dann: 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.
Voraussetzung ist die Kenntnis, dass dem *Window.mainMenu* ausschliesslich *MMenu*'s hinzugefügt werden können.

Weiter hatte ich folgendes Problem: beim Hinzufügen eines Fragments der Application, wurde nicht immer die Auswahl der Application angezeigt. Ein bisschen Fokus (des Feldes) verschieben und wieder reinklicken und manchmal die Space Taste drücken haben da geholfen.

Um Know-How über das Application Model aufzubauen ist der Live-Editor äusserst praktisch. Mit diesem kann das gesamte Application Model inspiziert werden.
Dazu fügt man beim Start der Applikation das Plugin *org.eclipse.e4.tools.emf.liveeditor* hinzu, der Live-Editor öffnet sich über den Shortcut ALT+SHIFT+F9.

Keine Exceptions bei Fehlern

Es kam des Öfteren vor, dass etwas, wie zum Beispiel die Anzeige eines Icons, nicht geklappt hat und dies ohne Fehlermeldung geschah. Hier könnte jeweils ein Stacktrace helfen. Ein Beispiel: part.setIconURI(„falscherPfad“) → keine Fehlermeldung, icon wird nicht angezeigt.

Menu Key Binding

Ich habe es lange nicht geschafft im Fragment ein Key Binding zu erfassen. Das Problem ist, dass die Erfassungsmaske keinen Binding Context aus dem Application Model kennt, dieser kann aber auch nicht manuell erfasst werden. Wenn in der XMI-Datei selbst etwas geändert wird, bringt das auch nichts, da der Editor den Inhalt immer wieder überschreibt. Eine Erfassung ist also nicht möglich und ohne Erfassung eines Binding Context funktioniert es auch nicht. Die folgende Fehlermeldung wurde beim Erfassen eines neuen Binding Contexts angezeigt: org.eclipse.core.commands.common.NotDefinedException: Cannot get the

parent identifier from an undefined context. ch.sbb.rcsd.client.map.ui.bindingcontext.0. Ich habe schlussendlich einen Weg gefunden, wie es trotzdem funktioniert. Das Ganze ist im Projekthandbuch[16] beschrieben.

11.3.6. Erfahrungen

In diesem Kapitel werden Ereignisse erwähnt die stattgefunden haben, aber nicht unbedingt als Probleme angeschaut werden.

Es sind in dieser Iteration keine neuen Erfahrungen dazu gekommen.

11.3.7. Risikobeurteilung

In diesem Kapitel werden die Risiken anhand der Erkenntnisse in dieser Iteration neu beurteilt

Id	Risiko	Risikobehandlung
1	Migration wegen unbekannten technischen Problemen nicht möglich	Zum diesem Punkt gibt es keine neuen Ergebnisse
2	Zeitplan kann nicht eingehalten werden	Der Zeitplan wurde nicht eingehalten, er wurde wiederum – und dieses Mal mehrere Tage – überschritten. Es sind mehrere Massnahmen nötig, damit dies nicht mehr passiert und damit die Arbeit zu Ende geschrieben werden kann.
3	Funktionalität geht verloren	Auch mit diesem Migrationsschritt konnte kein Verlust an Funktionalität festgestellt werden.
4	Performanceeinbussen	Gegenüber Aspekt 1 hat sich hier nicht geändert, da nur eine bestimmte View migriert wurde. Hier ist die Perfomance nicht messbar
5	Verlust von Stabilität	Es konnte kein Verlust an Stabilität festgestellt werden.
6	Verlust von Usability	Zum diesem Punkt gibt es keine neuen Ergebnisse
7	Wartbarkeit nimmt ab	Mit dem Ansatz vom Application Model ist die Wartbarkeit gestiegen
8	Testbarkeit nimmt ab	Die Migration dieses Aspektes hat keinen Einfluss auf die Testbarkeit
9	Look and Feel wird von Anwendern nicht toleriert	Das Look and Feel hat keine Änderungen erfahren
10	NEU! Migration nicht mit vertretbarem Aufwand möglich	RCS hat extrem viele Menus, das ist manuell nicht migrierbar! Es muss ein Weg gefunden werden der die Migration von Command, Handlers, etc. automatisch unterstützt.

11.3.8. Massnahmen

Da in dieser Iteration doch einiges schief lief habe ich mit dem Betreuer eine Krisensitzung abgehalten und diverse Massnahmen besprochen.

Um die eingeplanten Zeiten einzuhalten, habe ich beschlossen, bis auf weiteres jeweils 2 komplette Arbeitstage pro Woche anstelle des einen einzusetzen. Diese Massnahme habe ich mit dem PO an der Arbeit abgesprochen und es wurde genehmigt. Dazu hab ich beschlossen 2 Tage meiner Ferien zu opfern, damit ich gegen das Ende der Arbeit noch ein bisschen Reserven habe.

Ich arbeite die einzelnen Issues wieder sequentiell ab, es wird kein „Issue-Switching“ mehr geben. Dies gilt insbesondere auch wenn – aus welchen Gründen auch immer - Panik aufkommen sollte.

Zukünftig plane ich alle Meetings mit ein, dazu gehören auch die Review Meetings mit Marc.

[16] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.pdf

11.3.9. Lessons learned

In dieser Iteration habe ich ganz klar den Fokus teilweise verloren. Dies ist aus einem Grunde geschehen: ich habe die sequentielle Abarbeitung der einzelnen Issues nicht eingehalten und bin zwischen den Tasks hin- und hergesprungen. Das hat zu unnötigen Kontextwechsel (Context switch) geführt. Ich musste mich jedes Mal – bei einem Taskwechsel - wieder in die Thematik des Tasks einarbeiten und habe so viel Zeit verloren. In dieser Iteration kam bei mir ein bisschen Panik auf, ich hatte plötzlich das Gefühl die Arbeit nicht rechtzeitig fertigstellen zu können. In dieser Panik habe ich dann angefangen diese Kontextwechsel vermehrt zu machen.
Die Lektion ist also: **sequentielles Abarbeiten** der Tasks/Issues **IMMER** – auch in Paniksituationen - einhalten.

11.3.10. Fazit

Das Thema „Migration der Key Bindings“ konnte erst in der Aufräumiteration bearbeitet werden. Ich habe lange keinen Weg gefunden ein Key Binding mit Binding Context usw. in einem Fragment zum Laufen zu bringen. Es wurde entschieden dieses Thema aus zeitlichen Gründen später anzugehen. In der Aufräum-Iteration habe ich nun einen Weg gefunden. Erst jetzt kann man den Aspekt als komplett bearbeitet betrachten.

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

11.4. Aspekt „Services“

Mit diesem Aspekt wurde das Thema „Services“ behandelt. Das Thema beinhaltet Standard-Services sowie auch eigene Services.
Mit dem erlangten 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.

Ursprünglich war geplant, in derselben Iteration auch das Thema „Eigene Extension Points“ zu behandeln. Anbetracht der verbleibenden Zeit wurde mit dem Betreuer beschlossen, dieses Thema ausfallen zu lassen und den Fokus auf die Ausarbeitung und Qualität der Dokumente zu legen.

11.4.1. 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

11.4.2. Dauer der Iteration

Die Iteration war ursprünglich in der Periode

15.07.2013 – 28.07.2013

geplant, tatsächlich gedauert hat sie vom

01.08.2013 – 08.08.2013

gedauert. Die Iteration wurde nach einem Meeting mit dem Betreuer gestoppt. Bevor am Thema „Eigene Extension Points“ weitergearbeitet werden kann, müssen zuerst alle Dokumente komplett aktualisiert werden. Die Iteration war also 7 Tage kürzer als geplant.

11.4.3. Resultate

Was wurde erreicht?

Es konnte ein Weg aufgezeigt werden, wie die Standard-Services und auch eigene Services in E3 und auch in E4 zur Verfügung gestellt werden. Es wurde aufgezeigt welche Standard-Services existieren. Weiter wurde ein Weg aufgezeigt wie ein eigener Service migriert werden kann.

Was wurde nicht erreicht

Wie oben beschrieben wurde das Thema „Eigene Extensions Points“ aus Zeitgründen nicht behandelt.

Gegenüberstellung E3 und E4

In der folgenden Tabelle werden diverse Themen zu „Services“ einander gegenübergestellt. Die Tabelle schafft einen Überblick über die Unterschiede von E3 und E4:

Thema	E3	E4
Service-Art	Etwas selbst gestricktes, in RCS zum Beispiel eigene Service-Registry	OSGi-Services
Holen der Service Instanzen	Über IServiceLocator welcher von der Workbench oder WorkbenchWindow etc. implementiert wird. In RCS über statischen Methodenaufruf auf Services	Über Dependency Injection, OSGi Service-Instanzen werden im OSGi-Kontext abgelegt.
Eigene Implementationen für Standard-Services	Nein, nicht möglich	Ja, möglich
Anzahl Standard-Services	Viele	Weniger, Stichwort: The 20 things

Beurteilung Migration

Die Migration der Standard-Services ist gegeben. Ein Anwender in E4 nützt die neuen Services. Die Schwierigkeit der Migration von eigenen Services kann - je nachdem wie diese implementiert sind – ziemlich unterschiedlich sein. Bei RCS ist die Migration relativ leicht machbar, da durchgängig mit Interfaces gearbeitet wurde.

Auch hier stellt sich wieder die Frage, ob die RCS-Services nach E4 mit den geeigneten Tools automatisch migrieren lassen. Dies müsste man zumindest prüfen, würde aber den Zeitrahmen dieser Arbeit sprengen.

11.4.4. Test

Im Folgenden werden die durchgeführten Testfallszenarien dokumentiert.

Bezeichnung	Testfall A4_1		
Beschreibung	Prüfung Injektion AuthenticationService		
Voraussetzungen	Applikation ist im Debug Modus aus Eclipse gestartet, entsprechend **berechtigter** Benutzer ist eingeloggt, Breakpoint in *OpenMapViewHandler.isEnabled* gesetzt		
Begründung für Test	Mit dem Einbau des Authentications-Service soll getestet werden, ob dieser auch injiziert wird		
Fokus	Funktionalität		
Test-Datum	06.08.2013		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	In Hauptfenster Menu Fenster anklicken öffnen	Parameter authenticationService nicht *null*	Parameter authenticationService nicht *null*
Testergebnis	OK		

Bezeichnung	Testfall A4_2		
Beschreibung	Öffnen der geografischen Karte **mit** Berechtigung		
Voraussetzungen	Applikation ist gestartet, entsprechend **berechtigter** Benutzer ist eingeloggt		
Begründung für Test	Mit dem Einbau des Authentications-Service soll getestet werden, ob dieser auch aufgerufen wird und dies zum richtigen Resultat führt.		
Fokus	Funktionalität		
Test-Datum	06.08.2013		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	In Hauptfenster Menu Fenster anklicken öffnen	Menu Geografische Karte erscheint enabled	Menu Geografische Karte erscheint enabled
2	Geografische Karte Menupunkt auswählen	Menu Geografische Karte auswählbar Geografische Karte öffnet sich	Menu Geografische Karte auswählbar Geografische Karte öffnet sich
Testergebnis	OK		

Bezeichnung	Testfall A4_3		
Beschreibung	Öffnen der geografischen Karte **ohne** Berechtigung		
Voraussetzungen	Applikation ist gestartet, entsprechend **nicht berechtigter** Benutzer ist eingeloggt		
Begründung für Test	Mit dem Einbau des Authentications-Service soll getestet werden, ob dieser auch aufgerufen wird und dies zum richtigen Resultat führt.		
Fokus	Funktionalität		
Test-Datum	06.08.2013		
Schritt	**Bezeichnung**	**Erwartetes Ergebnis**	**Testergebnis**
1	In Hauptfenster Menu Fenster anklicken öffnen	Menu Geografische Karte erscheint disabled	Menu Geografische Karte erscheint disabled
Testergebnis	OK		

Testfall A4_1: Wir sehen, es wird tatsächlich eine Instanz des *ch.sbb.rcsd.client.auth.internal.AuthenticationService* injiziert.

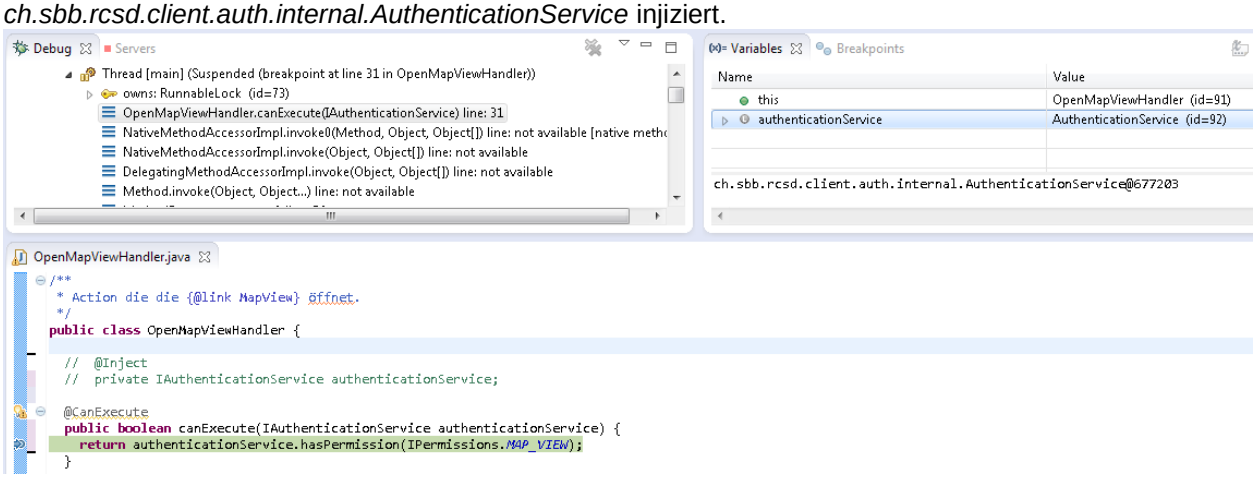

Abbildung 5 Debug Parameter AuthenticationService

Testfall A4_2 und Testfall A4_3: Es sollte die Applikation einmal als berechtigter Benutzer und einmal als nicht berechtigter gestartet werden. Wie erwartet war der Menupunkt wenn ich als berechtigter Benutzer eingeloggt habe enabled und sonst disabled:

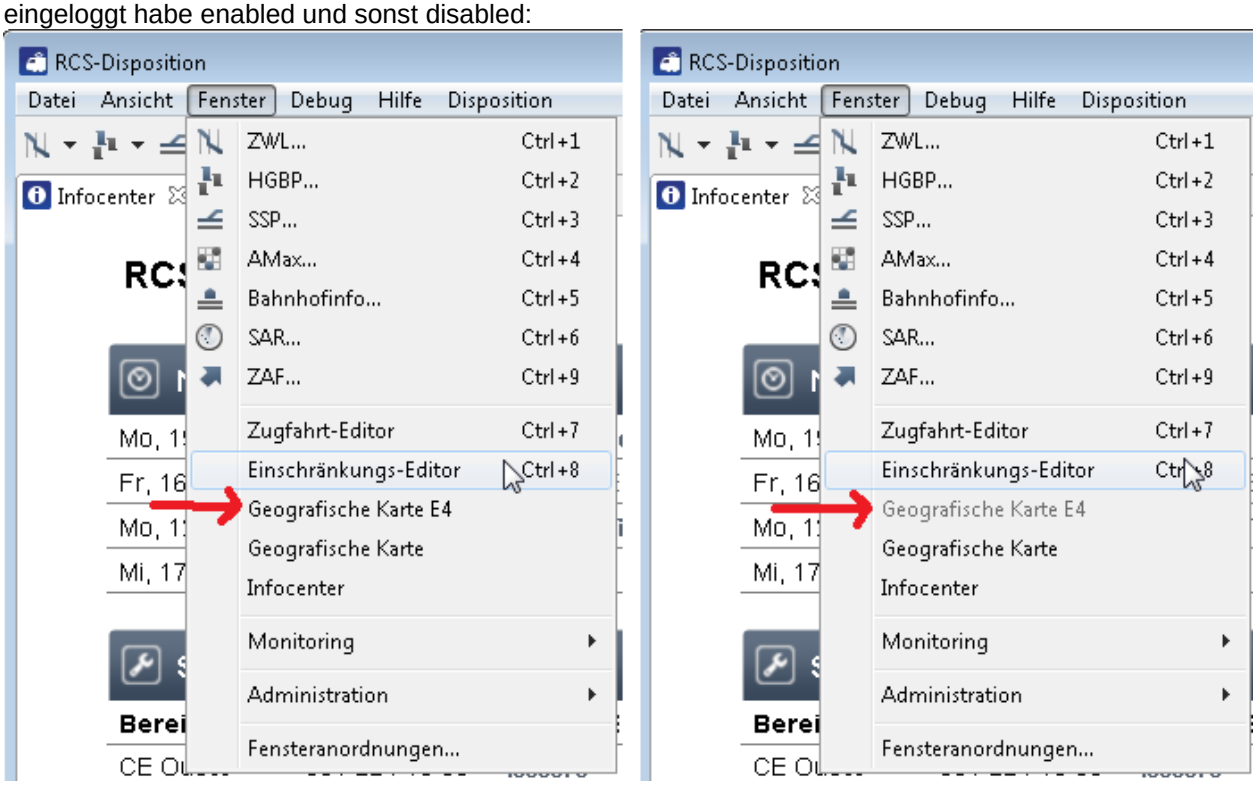

Abbildung 6 Test AuthenticationService Menupunkt enabled und disabled

11.4.5. Probleme

In diesem Kapitel wird beschrieben, welche Probleme aufgetreten sind, die das Voranschreiten der Arbeiten behindert haben. Auch weitere Probleme werden hier aufgelistet.

Service wird nicht instanziiert, injiziert

Es hat einige Zeit gedauert bis der Authentication Service in der Klasse *OpenMapViewHandler* injiziert wurde. Damit es funktioniert, müssen sämtliche Schritte, die im Projekthandbuch[17] aufgeführt sind, durchgeführt werden.

Keine Fehlermeldung bei Injizierung auf Methode

Bei der Injizierung des Services als Methodenparameter wurde keine Fehlermeldung ausgegeben. Im Gegensatz dazu wird bei der Injizierung auf ein Feld eine Fehlermeldung ausgegeben.

11.4.6. Erfahrungen

In diesem Kapitel werden Ereignisse erwähnt die stattgefunden haben, aber nicht unbedingt als Probleme angeschaut werden.

Zu diesem Punkt gibt es nichts zu ergänzen.

[17] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.pdf

11.4.7. Risikobeurteilung

In diesem Kapitel werden die Risiken anhand der Erkenntnisse in dieser Iteration neu beurteilt

Id	Risiko	Risikobehandlung
1	Migration wegen unbekannten technischen Problemen nicht möglich	Dieses Risiko hat sich gegenüber dem letzten Aspekt nicht verändert.
2	Zeitplan kann nicht eingehalten werden	Der Zeitplan ist eingehalten worden. Mit der Entscheidung jetzt mit den Aufräumarbeiten – zuungunsten des Teilaspektes „Eigene Extension Points“ zu starten, hat sich dieses Risiko wieder verkleinert. Das Projekt ist wieder auf Kurs.
3	Funktionalität geht verloren	Auch mit diesem Migrationsschritt konnte kein Verlust an Funktionalität festgestellt werden.
4	Performanceeinbussen	Gegenüber Aspekt 1 hat sich hier nicht geändert, da nur eine bestimmte View migriert wurde. Hier ist die Perfomance nicht messbar
5	Verlust von Stabilität	Es konnte kein Verlust an Stabilität festgestellt werden.
6	Verlust von Usability	Es konnte kein Verlust an Usability festgestellt werden.
7	Wartbarkeit nimmt ab	Dieses Risiko hat sich gegenüber dem letzten Aspekt nicht verändert.
8	Testbarkeit nimmt ab	Die Migration dieses Aspektes hat keinen Einfluss auf die Testbarkeit
9	Look and Feel wird von Anwendern nicht toleriert	Das Look and Feel hat keine Änderungen erfahren.
10	Migration nicht mit vertretbarem Aufwand möglich	Hier stellt sich auch die Frage nach der automatisierten Migration. RCS hat relativ viele eigene Services im Einsatz. Da wäre es praktisch über eine automatische Generierung von OSGi-Services zu verfügen.

Weitere, neue Risiken konnten nicht festgestellt werden.

11.4.8. Massnahmen

Ab sofort stehen Aufräumarbeiten an. Alle offenen Themen werden hinten angestellt und – falls noch Zeit bleibt – später behandelt. Die Aktualisierung aller Dokumente hat höchste Priorität.

11.4.1. Lessons learned

In dieser Iteration habe ich mich wieder an die sequentielle Abarbeitung der Tasks gehalten. Ich bin deswegen wieder um einiges produktiver geworden.
Ich habe vermehrt Tasks aufgeteilt, und kleinere Aufgaben gelöst. Diese Massnahme hat dazu geführt, dass die einzelnen Aufgaben leichter lösbar waren. Die grössere Anzahl an Task führte zu mehr kleinen Erfolgserlebnissen (beim Abschliessen eines Tasks). Dies hatte einen enormen positiven Einfluss auf meine Motivation.

Die Lektion dieser Iteration: **Teile und herrsche**.

11.4.2. Fazit

Die Iteration kann leider nur zur Hälfte als erfolgreich betrachtet werden. Das Thema „Eigene Extensions Points“ konnte nicht behandelt werden. Dafür konnte das Thema Services erfolgreich abgeschlossen werden.

Los geht’s mit der Aufräum-Iteration.

12. Aspekt „Application Model vs. Advisors“

12.1. Dauer der Iteration

Die Iteration war ursprünglich in der Periode

29.07.2013 – 11.08.2013

geplant, tatsächlich wurde sie aber aufgrund des Zeitmangels geopfert.

Das Thema ist jedoch teilweise durch den Aspekt 1 - in dem das Application Model erklärt wird - abgedeckt.

13. Aufräum-Iteration

Nach dem Review vom 08.08.2013 mit dem Betreuer wurde entschieden, die Aufräum-Iteration zu starten. 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.

Die Aufräumarbeiten betreffen das Projekthandbuch und den Projektbericht. In dieser Phase soll bestimmt werden, welche Themen im Projektbericht und welche im Projekthandbuch dokumentiert werden sollen.

Die Dokumente müssen in den folgenden Bereichen komplett aktualisiert werden:

- Formatierung
- Strukturierung
- TODOs bearbeiten
- Vergessene Teilaspekte einarbeiten

Des Weiteren sollen in dieser Iteration die Vorbereitung auf das Schlussreview, sowie das Schlussreview selbst stattfinden.

13.1. Dauer der Iteration

Die Iteration war ursprünglich in der Periode

12.08.2013 – 12.09.2013

geplant, tatsächlich gedauert hat sie vom

09.08.2013 – 12.09.2013

13.2. Resultate

In dieser Iteration konnten die beiden Dokumente Projekthandbuch[18] und Projektbericht[19] in der geforderten Qualität fertiggestellt werden.

13.3. Fazit

Diese Iteration hat gezeigt, dass Aufräumarbeiten enorm viel Aufwand generieren. Der Entscheid diese Iteration so früh zu starten war klug.

[18] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.pdf

[19] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projektbericht.pdf

14. Reviews

Es muss mindestens einen Zwischenreview-Termin und einen Schlussreview-Termin geben. Diese Termine fanden statt und sind in den folgenden Kapiteln dokumentiert.

14.1. Zwischenreview

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

- RCS, als das Rail Control System der SBB
  - Was ist RCS?
  - Wie sehen die UIs aus?
- Projekthandbuch
  - Struktur
  - Inhalt

14.2. Schlussreview

Am 05.09.2013 wurde mit dem Experten und dem Betreuer an der Haslerstrasse 30 in Bern das Schlussreview durchgeführt. In diesem Meeting wurde folgendes präsentiert:

- RCS läuft auf E4
- Die auf E4 migrierte Geografische Karte
- Projektbericht
- Projekthandbuch
- Restliche Arbeiten
  - Dokumente fertig stellen
  - Poster
  - Dokumentenuploads
  - Präsentation vorbereiten, halten

Am Schluss des Meetings gab es noch die Möglichkeit Fragen zu klären.

15. Präsentation

Die Arbeit wird am 20.09.2013 um 16:00 Uhr mit einer Präsentation abgeschlossen.
Die Präsentation dauert 10-15 Minuten, anschliessend bleiben 10-15 Minuten um Fragen zu beantworten. Insgesamt dürfen aber 25 Minuten nicht überschritten werden. Weitere Vorgaben zur Präsentation sind unter http://www.ti.bfh.ch/fileadmin/data/weiterbildung/SWS/Master_Thesis/9_Schlusspraesentation.pdf zu finden.

Um die Präsentation einmal zu Testzwecken durchzuführen werde ich am 18.09.2013 diverse Leute aus dem Projekt RCS einladen und den Vortrag halten.

15.1. Beurteilungskriterien für Diplompräsentationen

Die folgenden Kriterien werden von der Berner Fachhochschule vorgegeben:
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 spürbar.*
- *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*

16. Rückblick

Wie in jedem Projekt gab es auch in diesem Hochs und Tiefs.

Während der Bearbeitung des Aspektes „Commands / Handler, Menus, Key Bindings“ habe ich zum Beispiel das Ziel aus den Augen verloren. Ich geriet in Panik und habe die Issues nicht mehr sequentiell abgearbeitet. Dies führte – zusammen mit anderen Gründen - dazu, dass diese Iteration wesentlich länger gedauert hat, als geplant. Aus diesem Fehler habe ich gelernt. Ab diesem Zeitpunkt habe ich die Issues ausschliesslich sequentiell abgearbeitet.

Wenn eine Aufteilung eines Issues auf kleinere Issues möglich war, habe ich die Issues je länger je mehr aufgeteilt. Dies führte zu einem besseren Arbeitsfluss.

Die Fehler die ich gemacht habe, habe ich jeweils analysiert und so schnell wie möglich korrigiert.

Die Hochs fanden ganz klar jeweils nach der Bearbeitung eines Aspektes statt. Es war ein schönes Gefühl einen Aspekt abschliessen zu können.

Die Möglichkeit auf GitHub Milestones und Issues zu erfassen erwies sich als sehr praktisch. Der Entscheid dieses Tool einzusetzen war richtig.

Es wurden über 160 Issues auf 14 Milestones aufgeteilt:

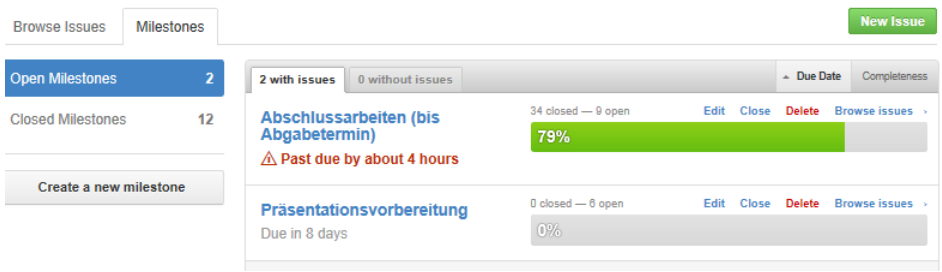

Abbildung 7 Letzter offener Milestone vor der Abgabe der Dokumente

Für mich ist das Projekt erfolgreich umgesetzt.

17. Zusammenfassung der Ergebnisse

Für die folgenden Themen konnten Wege zur Migration von E3 nach E4 aufgezeigt werden:

- Mixing E3/E4
- Dependency Injection/Adapters
- Commands / Handler, Menus, Key Bindings
- Services

Die Bearbeitung dieser Themen hat gezeigt, dass es möglich ist eine Applikation wie RCS von Eclipse RCP 3 nach Eclipse RCP 4 zu migrieren.

Es muss jedoch beachtet werden, dass mit dieser Arbeit aus Zeitgründen nicht alle Punkte - die bei einer Migration beachtet werden müssen – bearbeitet werden konnten. Es sind dies zum Beispiel die Migration eigener Extension Points, die automatisierte Migration von UI-Elementen oder die Perfomance-Probleme mit E4.

Um einen definitiven und gewissenhaften Entscheid über eine Migration von RCS zu fällen, müssten diese Punkte zuerst geklärt werden.

In der folgenden Aufstellung wird aufgezeigt, welche Argumente aus heutiger Sicht für oder gegen eine Migration von Eclipse RCP 3 nach Eclipse RCP 4 sprechen. Einige Punkte werden stark von anderen Punkten beeinflusst, wie zum Beispiel weniger Code führt zu einer sinkenden Fehleranfälligkeit..

Pro:

- Schlankere API
- Modernes Programmiermodell
- Steile Lernkurve
- Bessere Testbarkeit und Wartbarkeit
- Bessere Wiederverwendbarkeit
- Bessere Flexibilität und Erweiterbarkeit
- Fehleranfälligkeit sinkt
- E4 als langfristige Plattform
- Weniger Code nötig
- Lose Kopplung

Contra:

- Schlechtere Performance
- Automatisierte Migrationsmöglichkeit?
- Wenig Dokumentation
- Tooling im Bereich „Fragmente editieren“ nicht ausgereift
- Tooling im Bereich DI noch nicht ausgereift
- Der Aufwand für die Migration ist relativ gross

Die am Anfang der Arbeit aufgestellte These

> *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.*

hat sich bewahrheitet.

18. Verzeichnisse / Quellen

18.1. Abbildungsverzeichnis

18.2. Quellverzeichnis

Das Quellverzeichnis befindet sich im Projekthandbuch[20]

19. Glossar

Begriff	Definition
API	*Eine Programmierschnittstelle (englisch application programming interface, API; deutsch Schnittstelle zur Anwendungsprogrammierung) ist ein Programmteil, der von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird*[21]
Betriebspunkt	Der Betriebspunkt (BP) ist ein lokalisierbarer Bereich mit besonderen betrieblichen Anlagen wie beispielsweise Bahnhöfe, Haltestellen oder Spurnetze
Branch	Stammt aus der Versionsverwaltung. Entwicklungszweig eines Projektes[22]
Commit	*Commit ist ein Ausdruck aus der Softwaretechnik, der die bestätigende Freischaltung einer Änderung beschreibt* [23]
deprecated	Bezeichnet in der Softwareenwicklung Klassen oder auch Projekte die nicht mehr eingesetzt werden sollen, da sie eventuell bald entfernt werden
E4	In den Dokumenten als Synonym für Eclipse RCP 4
E3	In den Dokumenten als Synonym für Eclipse RCP 3
Eclipse RCP	Eclipse Rich Client Plattform, *Eclipse bietet die Rich Client Platform, welche es Anwendungsentwicklern ermöglicht, basierend auf dem Eclipse Framework, von der Eclipse-IDE unabhängige Anwendungen zu schreiben*[24]
Eclipse IDE	*Eclipse ist ein quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedenster Art*[25]

[20] https://github.com/MikeR13/MAS/blob/master/Deliverables/Projekthandbuch.pdf

[21] http://de.wikipedia.org/wiki/Programmierschnittstelle

[22] http://de.wikipedia.org/wiki/Versionsverwaltung#Branch

[23] http://de.wikipedia.org/wiki/Commit

[24] http://de.wikipedia.org/wiki/Eclipse_(IDE)

[25] http://de.wikipedia.org/wiki/Eclipse_(IDE)

Java	*Java ist eine objektorientierte Programmiersprache und eine eingetragene Marke des Unternehmens Sun Microsystems (2010 von Oracle aufgekauft). Die Programmiersprache ist ein Bestandteil der Java-Technologie – diese besteht grundsätzlich aus dem Java-Entwicklungswerkzeug (JDK) zum Erstellen von Java-Programmen und der Java-Laufzeitumgebung (JRE) zu deren Ausführung. Die Laufzeitumgebung selbst umfasst die virtuelle Maschine (JVM) und die mitgelieferten Bibliotheken.*[26]
Java Virtual Machine	Die Java Virtual Machine (abgekürzt Java VM oder JVM) ist der Teil der Java-Laufzeitumgebung (JRE) für Java-Programme, der für die Ausführung des Java-Bytecodes verantwortlich ist. Hierbei wird im Normalfall jedes gestartete Java-Programm in seiner eigenen virtuellen Maschine (VM) ausgeführt. Der andere Teil der Java-Laufzeitumgebung sind die Java-Klassenbibliotheken[27]
JUnit	JUnit ist ein Framework zum Testen von Java-Programmen, das besonders für automatisierte Unit-Tests einzelner Units (Klassen oder Methoden) geeignet ist[28]
(G)UI	*Eine grafische Benutzeroberfläche (GBO) ist eine Software-Komponente, die dem Benutzer eines Computers die Interaktion mit der Maschine über grafische Symbole erlaubt.*[29]
Singleton	*Das Singleton (selten auch Einzelstück genannt) ist ein in der Softwareentwicklung eingesetztes Entwurfsmuster und gehört zur Kategorie der Erzeugungsmuster (englisch creational patterns). Es stellt sicher, dass von einer Klasse genau ein Objekt existiert.[1] Dieses Singleton ist darüber hinaus üblicherweise global verfügbar*[30]
POJO	*POJO ist eine Abkürzung für Plain Old Java Object, also ein „ganz normales" Objekt in der Programmiersprache Java*[31]
Repository	*Ein Repository (engl. für ‚Lager', ‚Depot' oder auch ‚Quelle', pl. Repositories), auch - direkt aus dem Lateinischen entlehnt - Repositorium (pl. Repositorien) , ist ein verwaltetes Verzeichnis zur Speicherung und Beschreibung von digitalen Objekten*[32] Im Zusammenhang mit dem Projekt sind die digitalen Objekte Java Sourcecode Dateien.
Subversion	*Apache Subversion (SVN) ist eine freie Software zur Versionsverwaltung von Dateien und Verzeichnissen*[33]
Targetplattform	Zielplattform zum Entwickeln von Software.
Trunk	Stammt aus der Versionsverwaltung. Hauptentwicklungszweig eines Projektes[34]

[26] http://de.wikipedia.org/wiki/Java_(Programmiersprache)

[27] http://de.wikipedia.org/wiki/Java_Virtual_Machine

[28] http://de.wikipedia.org/wiki/JUnit

[29] http://de.wikipedia.org/wiki/GUI

[30] http://de.wikipedia.org/wiki/Singleton_(Entwurfsmuster)

[31] http://de.wikipedia.org/wiki/Plain_Old_Java_Object

[32] http://de.wikipedia.org/wiki/Repository

[33] http://de.wikipedia.org/wiki/Subversion_(Software)

[34] http://de.wikipedia.org/wiki/Versionsverwaltung