



# Grafische Einbindung von Plug-ins im Eclipse Rich Client Platform Umfeld

**Praxisbericht**

des Studiengangs Angewandte Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

**Max Emmert**

Februar 2014


| | |
|---|---|
| **Matrikelnummer, Kurs** | 8132098, STG-TINF12C |
| **Ausbildungsfirma** | Compart AG, Böblingen |
| **Betreuer** | Lars Heppler |

# Sperrvermerk

Die vorliegende Praxisbericht mit dem Titel *Grafische Einbindung von Plug-ins im Eclipse Rich Client Platform Umfeld* ist mit einem Sperrvermerk versehen und wird ausschließlich zu Prüfungszwecken am Studiengang Angewandte Informatik der Dualen Hochschule Baden-Württemberg Stuttgart vorgelegt. Jede Einsichtnahme und Veröffentlichung – auch von Teilen der Arbeit – bedarf der vorherigen Zustimmung durch die Compart AG.

# Erklärung

Ich erkläre hiermit ehrenwörtlich:

1. dass ich meinen Praxisbericht mit dem Thema *Grafische Einbindung von Plug-ins im Eclipse Rich Client Platform Umfeld* ohne fremde Hilfe angefertigt habe;
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe;
3. dass ich meine Praxisbericht bei keiner anderen Prüfung vorgelegt habe;

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Stuttgart, Februar 2014

Max Emmert

## Zusammenfassung

**Thema der Arbeit**
Einbindung von Plug-ins im Eclipse RCP[1]-Umfeld, sowie deren grafische Einbettung in eine bestehende RCP-Anwendung auf Basis von OSGi[2].
**Stichworte**
OSGi, Eclipse RCP, GUI, Filterprofil, Plug-in
**Kurzzusammenfassung**
In dieser Arbeit wird untersucht, inwiefern es möglich ist, Plug-ins in eine bestehende Eclipse RCP-Applikation zu integrieren. Dabei steht vor allem die grafische Einbindung in das bestehende Softwarekonstrukt im Vordergrund. Die Konzeption und Umsetzung eines Plug-ins im RCP-Umfeld gehört ebenfalls zum Kern dieser Arbeit. Neben den Grundlagen der Eclipse RCP Entwicklung werden verwandte Themen, wie beispielsweise OSGi behandelt.

[1] Rich Client Platform
[2] Open Services Gateway initiative

## Summary

**Thema der Arbeit**
Integration of plug-ins in an existing Eclipse RCP environment, as well as their graphical embedding in an RCP application based on OSGi.
**Stichworte**
OSGi, Eclipse RCP, GUI, Filterprofil, Plug-in
**Kurzzusammenfassung**
In this work it is studied how to extent it is possible to integrate plug-in into an existing Eclipse RCP application. In particular, the graphic integration into the existing software construct is in the foreground. The design and implementation of a plugin in the RCP environment is also part of the core of this work. Besides the basics of Eclipse RCP development related issues, such as OSGi are treated.

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Motivation

Noch vor einigen Jahren war es nicht unüblich, Anwendungen über die Kommandozeile zu steuern. Die funktionalen Aspekte einer Anwendung hatten eine höhere Priorität als ihre Nutzbarkeit. Um komplizierte Applikationen über die Kommandozeile zu bedienen ist jedoch oft Expertenwissen notwendig. Deshalb spielt die Entwicklung von grafischen Oberflächen in der heutigen Anwendungsentwicklung eine immer größere Rolle. Es reicht oft nicht aus, dem Benutzer nur eine grafische Möglichkeit zur Steuerung einer Applikation zu geben. Im Rahmen der Software-Ergonomie spielt die Benutzerführung eine tragende Rolle. So auch bei der Anwendung Workbench for Mill Plus, eine Anwendung der Compart AG, die es dem Anwender ermöglicht bequem Prozesse zur Modifikation und Konvertierung von Dokumenten zu steuern. Der Anwendung Workbench for Mill Plus liegt die Programmiersprache Java zu Grunde. Durch die hohe Verbreitung von Java entstehen immer komplexere Anwendungen. Um die Komplexität solcher Anwendungen handhaben zu können bedient man sich oft Mitteln zur Modularisierung. Hierfür hat Java jedoch keine eigene Sprachunterstützung. OSGi bietet die Möglichkeit, monolithischen Anwendungen, die den heutigen dynamischen Anforderungen nicht gerecht werden, entgegenzuwirken.

## 1.2 Zielsetzung

In der vorausgegangenen Praxisphase wurde vom Studenten ein Programm entwickelt, das es ermöglicht Filterprofile anhand eines gegebenen XML-Schemas zu aktualisieren. Diese Funktionalität soll nun in die DocBridge Workbench for Mill Plus integriert werden. Dies würde Kunden bei der Auslieferung eines neuen XML-Schemas die Möglichkeit geben, die Konfiguration ihrer alten Profile beizubehalten beziehungsweise

ein neues Filterprofil zu erstellen, das die Konfiguration des alten Filterprofils enthält, jedoch zum neuen XML-Schema valide ist. Darüber hinaus soll es dem Anwender möglich sein, Profile in die DocBridge Workbench for Mill Plus zu importieren. Wird beim Import festgestellt, dass das Filterprofil nicht dem gegebenen XML-Schema entspricht soll dies dem Anwender kenntlich gemacht werden.

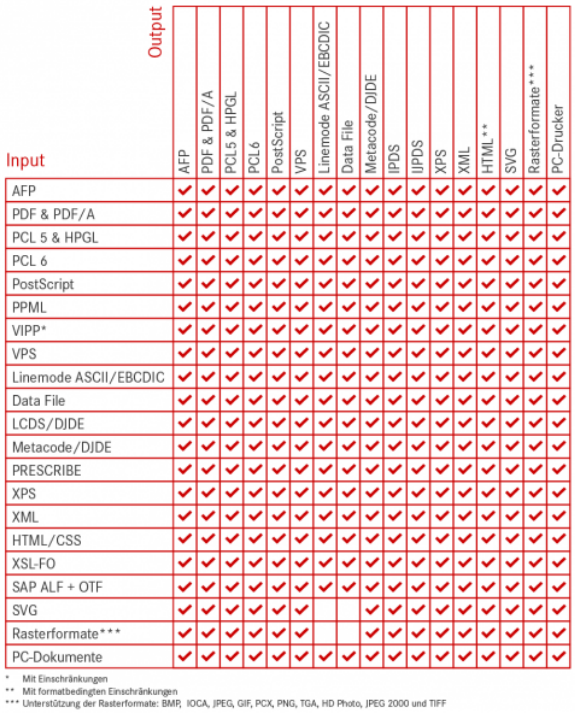
## 1.3 Aufbau der Arbeit

Die Einleitung soll dem Leser einen Überblick über das Thema verschaffen. Es werden sowohl die Motivation hinter dieser Arbeit als auch deren Zielsetzung abgehandelt. Das Kapitel Grundlagen soll dem Leser einen Überblick über das Thema verschaffen. Es werden die Motivation und die Zielsetzung dieser Arbeit erläutert. Darüber hinaus werden Begriffe und Systeme erklärt, die für das Verständnis und Nachvollziehbarkeit der Arbeit grundlegend sind. Im Hauptteil wird die Vorgehensweise zur Problemlösung detailliert beschrieben. Er umfasst die Analyse der zugrunde liegenden Thematik ebenso wie das Konzept zur Lösung der behandelten Aufgabe. Die Umsetzung wird im Abschnitt Implementierung beschrieben. Im Abschluss daran wird auf die Ergebnisse der Arbeit eingegangen.

# 2 Grundlagen

## 2.1 DocBridge Mill Plus

DocBridge Mill Plus ist eine plattformunabhängige, skalierbare Software für die Anzeige und Verarbeitung von Datenströmen. Sie ermöglicht die Analyse und Modifikation von Dokumenten in verschiedensten Formaten und die Konvertierung in unterschiedliche Ausgangsformate. Die Dokumente lassen sich auf nahezu allen gängigen physikalischen und digitalen Kanälen darstellen. Mit DocBridge Mill Plus ist der Anwender auch in der Lage Prozesse, die im Output-Management üblich sind, abzubilden. Der modulare Aufbau der Software lässt es zu, sie mit zusätzlichen Modulen um Ein-und Ausgabeformate und Workflows zu erweitern.



| Input \ Output | AFP | PDF & PDF/A | PCL5 & HPGL | PCL6 | PostScript | VPS | Linemode ASCII/EBCDIC | Data File | Metacode/DJDE | IPDS | IJPDS | XPS | XML | HTML** | SVG | Rasterformate*** | PC-Drucker |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AFP | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PDF & PDF/A | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PCL 5 & HPGL | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PCL 6 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PostScript | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PPML | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| VIPP* | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| VPS | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Linemode ASCII/EBCDIC | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Data File | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| LCDS/DJDE | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Metacode/DJDE | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PRESCRIBE | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| XPS | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| XML | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| HTML/CSS | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| XSL-FO | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| SAP ALF + OTF | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| SVG | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Rasterformate*** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| PC-Dokumente | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

* Mit Einschränkungen
** Mit formatbedingten Einschränkungen
*** Unterstützung der Rasterformate: BMP, IOCA, JPEG, GIF, PCX, PNG, TGA, HD Photo, JPEG 2000 und TIFF

Abbildung 2.1: Mögliche Verarbeitungsformate der DocBridge Mill Plus

## 2.2 DocBridge Workbench for Mill Plus

DocBridge Workbench for Mill Plus bietet eine grafische Benutzeroberfläche für die Filterkonfiguration und die formatunabhängige Dokumentendarstellung. In der DBWB[1] können einzelne, mehrere oder in Stapeln zusammengefasste Dokumente angezeigt werden. Es werden alle in Abbildung 2.1 aufgeführten Formate unterstützt. Geladene Dokumente können in der, in Abbildung 2.3 vorgestellten Dokumentenperspektive einheitlich dargestellt werden. Alle Dokumente, die in einem der unterstützten Formaten vorliegen, können im Anzeigebereich untersucht werden. So ist es dem Nutzer möglich, sich unabhängig vom Format des Dokuments, völlig auf dessen Inhalt des zu konzentrieren. In der Prozessperspektive (Abbildung 2.2) können Verarbeitungsschritte konfiguriert, validiert und lokal ausgeführt werden. Die erstellte Prozesskonfiguration kann exportiert und in der DocBridge Mill Plus ausgeführt werden. Die Konfiguration der Prozesse erfolgt überwiegend über Drag & Drop. Die Prozessperspektive ist die Standartperspektive der DBWB.

[1] DockBridge Workbench

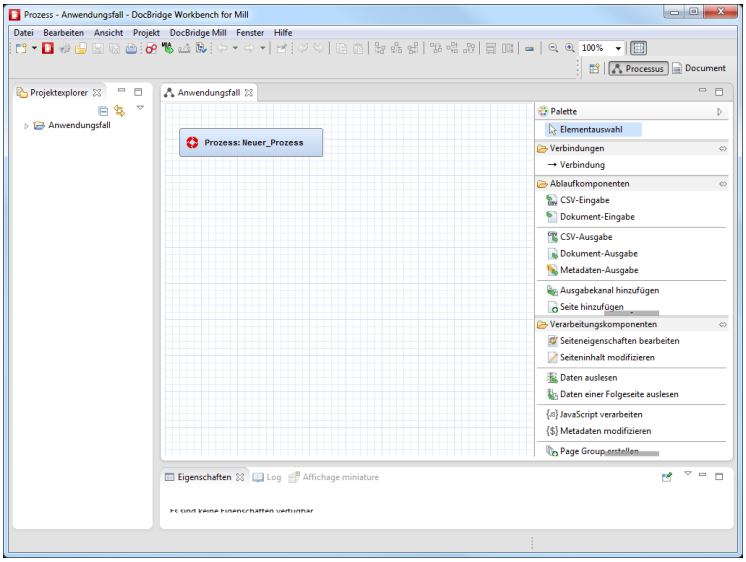



Abbildung 2.2: Prozessperspektive

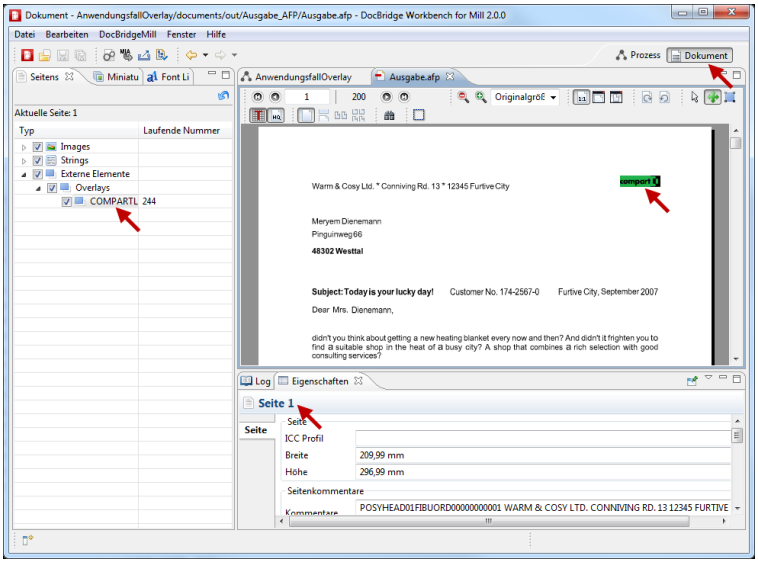



Abbildung 2.3: Dokumentperspektive

## 2.3 OSGi

OSGi Services ermöglichen die Zusammenarbeit einzelner Bundles. Java stellt nur begrenzte Möglichkeiten zur Modularisierung bereit. Java Klassen lassen sich zwar unter der Verwendung von Packages gruppieren, jedoch ist es nicht möglich diese Gruppierungen und Restriktionen zur Laufzeit beizubehalten. Die Möglichkeit, einzelne Klassen als package-protected zu deklarieren ist meist keine befriedigende Lösung. OSGi ermöglicht die Verwaltung der Sichtbarkeit von Packages zur Laufzeit. OSGi ist ein Framework, welches sich auch dem Problem der Modularisierung annimmt und die Entwicklung von modularen Anwendungen auf der Java-Plattform ermöglicht, was in Abbildung 2.4 veranschaulicht wird. OSGi-Anwendungen bestehen aus einzelnen Modulen, die auch Bundles genannt werden. Jedes Bundle hat einen eigenen Lebenszyklus. Dies bedeutet, dass zur Laufzeit einzelne Bundles dynamisch geladen und entfernt werden können. Die Spezifikation von OSGi wird von der OSGi Alliance erstellt. Die OSGi Alliance wurde 1999 von rund 30 namhaften Firmen gegründet, um den Anforderungen im Bereich der eingebetteten Systeme gerecht zu werden. (Kriens, 2008, S.3) Die Eclipse IDE[1] arbeitet seit Version 3.0 mit der OSGi Implementierung Equinox.

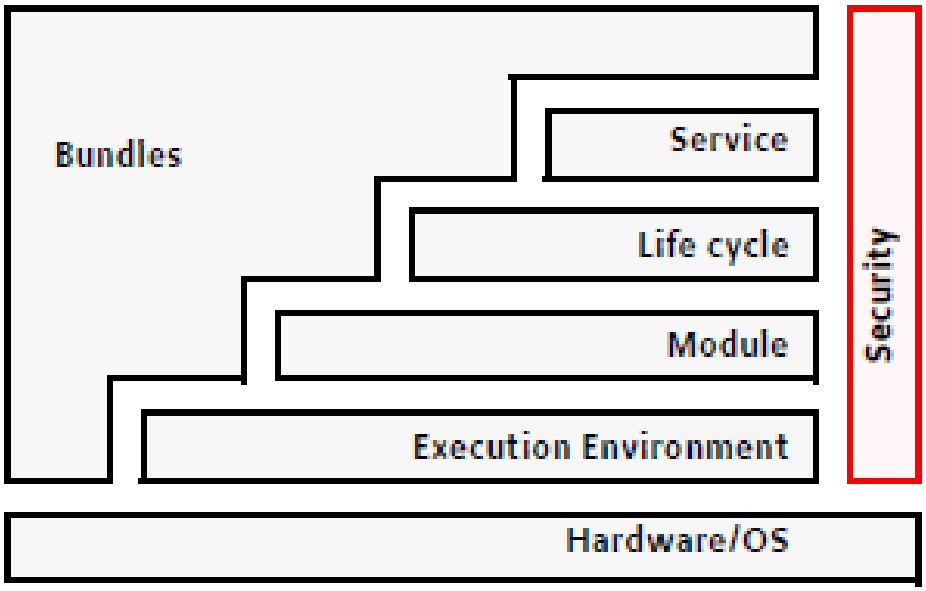



Abbildung 2.4: OSGi Schichtenmodell (OSGiAlliance, 2012, aus S.2)

[1] Integrierte Entwicklungsumgebung

## 2.4 OSGi Services

Bei OSGi Services handelt es sich um Java-Objekte, die unter einem Interface in der OSGi Service-Registry angemeldet werden. In der Service-Registry angemeldete Services können von anderen Bundles verwendet werden. Services können genau wie OSGi-Bundles dynamisch geladen und entfernt werden. Wird ein Bundle, das einen Service bereitstellt gestoppt, so muss das verwendende Bundle auf das Wegfallen des Service entsprechend reagieren. (OSGiAlliance, 2012, vgl.S.114)

## 2.5 SWT

Das SWT[1] stellt das Framework für grafische Oberflächen auf der Eclipse-Plattform bereit. SWT stellt eine Java Schnittstelle für die nativen GUI-Bibliotheken verschiedener Betriebssysteme zur Verfügung. Hierzu gehören Microsoft Windows, Linux[2] und Mac OS X[3]. Da auf die spezifischen Steuerelemente des jeweiligen Betriebssystems zugegriffen wird, kann mit SWT auf den oben genannten Plattformen ein natives Aussehen und Verhalten erzielt werden, wie in Abbildung 2.5 veranschaulicht wird.

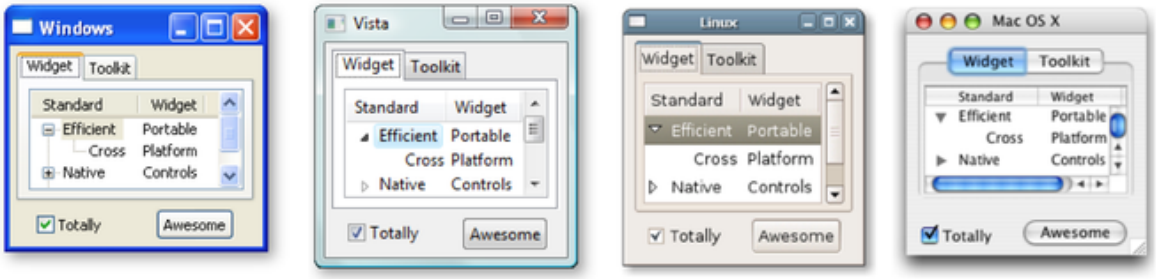



Abbildung 2.5: Darstellung von SWT-Widgets auf verschiedenen Betriebssystemen[4]

## 2.6 Eclipse RCP

Eclipse RCP ist eine Plattform zur Entwicklung von Desktop-Anwendungen. Sie entstand aus der Eclipse IDE, die sich aufgrund ihrer allgemeinen Natur anbietet. Das

1 Standart Widget Toolkit
2 GIMP Toolkit
3 Cocoa
4 aus (Ebert, n.d.)

Eclipse RCP Grundgerüst kann durch eigene Anwendungsfunktionalitäten erweitert werden. Viele der vorhandenen Komponenten können in eigenen Anwendungen im Workbench- Aufbau genutzt werden. Als Beispiel wären hier Komponenten für eine Hilfe-Funktion oder das erweiterbare Plug-in-System zu nennen. Vorteile der Anwendungsentwicklung mit Hilfe von Eclipse RCP sind Grundstrukturen für den Aufbau von GUI-Anwendungen und der modulare Aufbau. Dieser ermöglicht es, dass Erweiterungen auf Eclipse RCP Basis miteinander harmonieren. Die zum implementieren benötigten Werkzeuge sind bereits in der Eclipse IDE (Eclipse for RCP/Plug-in Developers)bereits enthalten.

## 2.7 Target Platform

Eclipse RCP Applikationen werden für eine konfigurierbare Zielplattform[1] entwickelt. Alle externen Abhängigkeiten werden über die Target Platform geladen. Soll mit Plug-ins die Eclipse IDE selbst erweitert werden, verwendet man die standardmäßig die Eclipse IDE selbst als Target Platform. (Ebert, n.d., vgl.S14) Bei der Entwicklung einer eigenen Rich Client Platform, oder Plug-ins für eine solche, ist es nötig eine eigene Target Platform zu nutzen, da sonst alle Plug-ins und Funktionalitäten der Standard Eclipse IDE auch in der eigenen Applikation wiederzufinden wären. Durch die Definition der Target Platform kann festgelegt werden, in welcher Version der Applikation bestimmte Plug-ins eingebunden werden. Im Idealfall arbeiten alle Entwickler eines Projekts mit der selben Target Platform. So wird eine einheitliche Entwicklungsumgebung sichergestellt.

## 2.8 Plug-in zur Profilkonvertierung

Vom Studierenden wurde während der letzten Praxiseinheit ein Plug-in im OSGi-Umfeld entwickelt, welches dazu verwendet werden kann Filterprofile zu aktualisieren. Die Konfiguration von Konvertierungsabläufen wird in Filterprofilen festgehalten. Deren Aufbau ist in einem XML-Schema definiert. Kommen Funtionalitäten oder Komponenten hinzu, wird das Schema aktualisiert und somit abgeändert. Bestehende Filterprofile sind nun nicht mehr zum Schema valide. Das Plug-in zur Profilkonvertierung liest die Informationen aus dem alten Profil mit Hilfe von verschiedenen Java

[1] Target Platform

API[1]s aus und generiert anhand des Schemas ein neues Profil, das zum Schema valide ist. Es kommen vor allem die APIs Jsoup[2] und JaxB[3] zur Verwendung.

[1] Application Programming Interface
[2] API zum Arbeiten mit degeneriertem HTML
[3] API für das Erstellen von Klassen aus einer XML-Schema-Instanz

# 3 Hauptteil

## 3.1 Analyse

### 3.1.1 Anwendungsfälle

In diesem Abschnitt werden die verschiedenen Anwendungsfälle für die zu entwickelnde Grafische Oberfläche erläutert.

#### Profil importieren

Der Kunde soll in der Lage sein, ein Profil in seinem Datenspeicher über die Menüführung der Workbench for Mill Plus auszuwählen. Das ausgewählte Profil soll in der Workbench geöffnet und in einem neuen Editor zur weiteren Bearbeitung angezeigt werden.

#### Profil aktualisieren

Ist ein Profil im Editor der Workbench for Mill Plus geöffnet, soll es dem Anwender über verschiedene Methoden der Benutzerführung ermöglicht werden, die Aktualisierung des Profils zu starten. Ist das Profil zum aktuellen XML-Schema valide, soll keine Aktualisierung durchgeführt werden.

#### Importiertes Profil verwenden

Ein importiertes Profil soll nach seinem Import in der DBWB im Workflow des Benutzers verwendet werden können.

#### Erzeugtes Profil verwenden

Ein vom Aktualisierungsvorgang erzeugtes Profil soll nach seiner Erstellung in der DBWB im Workflow des Benutzers verwendet werden können.

### 3.1.2 Nichtfuntionale Anforderungen

#### Handhabung

Bei den Anwendern der DBWB handelt es sich meist um Personen, die im Dokumentmanagement tätig sind. Von Ihnen kann kein Programmiertechnisches Expertenwissen erwartet werden. Somit ist es erforderlich, die Funktionalität zur Aktualisierung von Profilen möglichst intuitiv in den bestehenden Workflow zu integrieren.

#### Wartbarkeit

Die einzelnen OSGi-Bundles sowie deren Zusammenspiel muss einfach wartbar sein, damit sie erfolgreich in das bestehende Produkt integriert werden können. Auch für Updates und zukünftige Entwicklungen ist eine gute Wartbarkeit von großer Bedeutung. Ist eine Software schlecht wartbar, lassen sich neue oder geänderte Anforderungen meist nur mit hohem Aufwand umsetzen.

#### Portierbarkeit

Das zu entwickelnde Plug-in soll auf allen Systemen verfügbar sein, auf denen auch die Workbench for Mill Plus eingesetzt wird. Die Portierbarkeit bei modular aufgebauten Systemen hat eine große Bedeutung. Wird ihr wenig Aufmerksamkeit geschenkt, wird das entwickelte System nicht auf allen Zielplattformen das gleiche Verhalten aufweisen können.

#### Konsistenz

Die Erweiterungen der DBWB sollen eine Konsistenz in der Benutzerführung aufweisen. Zusätzliche Module und Plug-ins sollen nahtlos in den bestehenden Arbeitsablauf

integriert werden, um die gewohnte Benutzbarkeit für den Anwender zu gewährleisten.

## 3.2 Konzept

In diesem Abschnitt wird das, dem zu entwickelnden Plug-in zugrundeliegende, Konzept erläutert.

### 3.2.1 Grafische Einbettung

Die neu bereitgestellten Funktionalitäten sollen sich nahtlos in die bestehende Anwendung DocBridge Workbench for Mill integrieren. Darüber hinaus soll die grafische Einbettung den Standards der Compart AG für GUI[1]s genügen. Hierzu wurden die Benutzeroberfläche der DBWB und firmeninterne Designvorlagen untersucht. Um eine hohe Benutzbarkeit zu gewährleisten werdem dem Anwender mehrere Möglichkeiten gegeben, auf die hinzugefügten Funktionalitäten zuzugreifen. Um die grafische Einbindung zu visualisieren und zu veranschaulichen wurden Mockups, funktionsunfähige Modelle, angefertigt.

### 3.2.2 Prototyp

Um die in den Anforderungen geforderten Funktionalitäten herzustellen werden diese erst in einem Prototypen simuliert. Dieser beinhaltet eine Komponente zum importieren und eine Komponente zum aktualisieren von Filterprofilen. Der Prototyp konzentriert sich bei der Aktualisierungskomponente zunächst auf den Dokumenttyp AFP[2].

#### Komponente für den Profilimport

Filterprofile werden in der DBWB in einem Profile Repository verwaltet. Für gewöhnlich wird mit jeder neuen Version eines XML-Schemas ein default Profil mitgeliefert, anhand dessen Vorlage sich der Anwender ein eigenes Profil erstellen lassen kann. Das erstellte Filterprofil kann beliebig modifiziert werden. Das default Profil ist nicht

[1] Graphical User Interface
[2] Apple Filing Protocol

zur Modifikation vorgesehen. Die Komponente liest aus dem importierten Profil die Version aus. Die Version ist in jedem Profil als Attributswert abgelegt. Anhand der Version und einem eindeutigen Identifier wird eine neue Version im Profile Repository angelegt. Das Profil wird als neues default Profil dieser Version abgelegt. Das hat den Vorteil, dass der Anwender sein importiertes Profil in der Dokumentenperspektive der DBWB untersuchen, jedoch nicht verändern kann. Große Bedeutung kommt auch der Benutzerführung zu. In jeder der beiden Perspektiven der DBWB ist der Import auf unterschiedliche Art und Weise zu handhaben.

**Prozessperspektive** Dem Anwender steht in dieser Perspektive eine Projektnavigation zur Verfügung. Er soll beim Import eine Auswahlmöglichkeit für das Zielprojekt haben. Importierte Profile werden in einem Unterordner 'profiles' im ausgewählten Projekt abgelegt. Anfangs war es vorgesehen, in der Prozessperspektive mehrere Filterprofile gleichzeitig importieren zu können. Während der Implementierung der Importkomponente wurde klar, dass dies der Konsistenz und Nutzerführung nicht zuträglich ist. Die Funktionalität wurde deshalb auf den Import eines einzelnen Profils beschränkt. Nach dem Import wird das Profil im Profile Repository abgelegt und in einem Editor geöffnet. Existiert im Projektunterverzeichnis 'profiles' bereits ein Profil für den entsprechenden Filtertyp soll der Nutzer die Möglichkeit haben, das alte Profil zu überschreiben oder seine Aktion abzubrechen.

**Dokumentenperspektive** In der Dokumentenperspektive betrachtet der Benutzer oft nur ein einzelnes Dokument, das mit Hilfe verschiedener Filter konvertiert werden kann. Deshalb ist es unnötig mehrere Profile importieren zu können. Die Komponente zur Projektnavigation ist in dieser Perspektive ebenfalls deaktiviert. Somit soll es beim Import in der Dokumentenperspektive auch nicht möglich sein, ein Projekt auszuwählen, in dem das Profil gespeichert wird. Es wird nur im Profile Repository abgelegt und in einem Editor geöffnet.

### Komponente zur Aktualisierung von Profilen

Bei der Aktualisierung von Profilen soll ein bereits vorhandenes Plug-in verwendet werden, das die Funktionalität der Konvertierung von Profilen zur Verfügung stellt. Geplant ist, dass für jeden Filtertyp ein eigenes Plug-in erstellt wird. Diese werden bei Bedarf eingebunden. Wie auch bei der Importkomponente soll sich die Aktualisierungskomponente in den bestehenden Arbeitsfluss integrieren. Ist ein Filterprofil in einem

Editor ausgewählt, hat der Anwender verschiedene Möglichkeiten den Konvertierungsprozess anzustoßen. Neben einem Eintrag im Hauptmenü soll die Aktion auch über eine Toolbar und ein Kontextmenü verfügbar sein. Nach starten des Prozesses über eine der genannten Möglichkeiten wird das aktuell geöffnete Profil ermittelt. Anhand des root-Tags[1] wird der Filtertyp(z.B. AFP,PDF[2]) ermittelt. Anhand dieses Filtertyps wird das benötigte Konvertierungs-Plug-in festgestellt und dessen convert-Methode mit dem ausgewählten Profil als Übergabeparameter gestartet. Sobald die Konvertierung abgeschlossen ist, wird dem Anwender das originale und das generierte Profil angezeigt, um weitere Modifikationen vornehmen zu können. Zur Vergleichsanzeige der Dateien wurden folgende Ansätze untersucht.

**Erweiterung des Dokumenteneditors**

Der Editor der DBWB ermöglicht es, eine Vorschau von zu konvertierenden Dokumenten anzuzeigen. Ist beispielsweise ein Dokument im AFP-Format gegeben und soll nach PDF konvertiert werden, so lässt sich das Ergebnis in einem Editor bereits zur Laufzeit betrachten. Geplant war das Überschreiben dieses Editors mit einem Standard XML-Editor. Dieser Ansatz musste verworfen werden, da dies im konzeptionellen Widerspruch zum Dokumenteneditor stand.

**Erstellen eines spezialisierten Editors**

Der nächste Ansatz war es, einen Editor speziell zum Vergleich von zwei Filterprofildateien zu entwickeln. Ein EditorPart soll zwei weitere EditorParts enthalten, in denen das originale und das konvertierte Profil angezeigt werden. Eclipse RCP sieht jedoch vor, dass ein Editor in Beziehung zu genau einer Datei steht. Deshalb musste dieser Ansatz ebenfalls verworfen werden.

**Nutzung von CompareUI[3]**

Die Java Klasse `org.eclipse.compare.CompareUI` bietet einen Einstiegspunkt, um konfigurierbare Vergleichsoperationen auf beliebige Ressourcen durchzuführen. Das Ergebnis eines Vergleichs kann in einem speziellen Vergleichseditor geöffnet werden. Dieser Editor ermöglicht das dynamische Vergleichen einzelner Elemente von Dokumenten, was besonders am Beispiel von XML[4] oder HTML[5] ersichtlich wird.

1 Hauptelement in der Profildatei
2 Portable Document Format
3 Java API um Vergleich von Objekten
4 Extensible Markup Language
5 Hypertext Markup Language

Auch das Eclipse Plug-in eGit[1] nutzt die CompareUI, um Dateien in der Versionierungshistorie zu vergleichen. Da die Vorteile dieser Technologie nicht von der Hand zu weisen sind, wird dieser Ansatz zur Entwicklung zur Darstellung des Ergebnisses des Konvertierungs-Plug-ins gewählt.

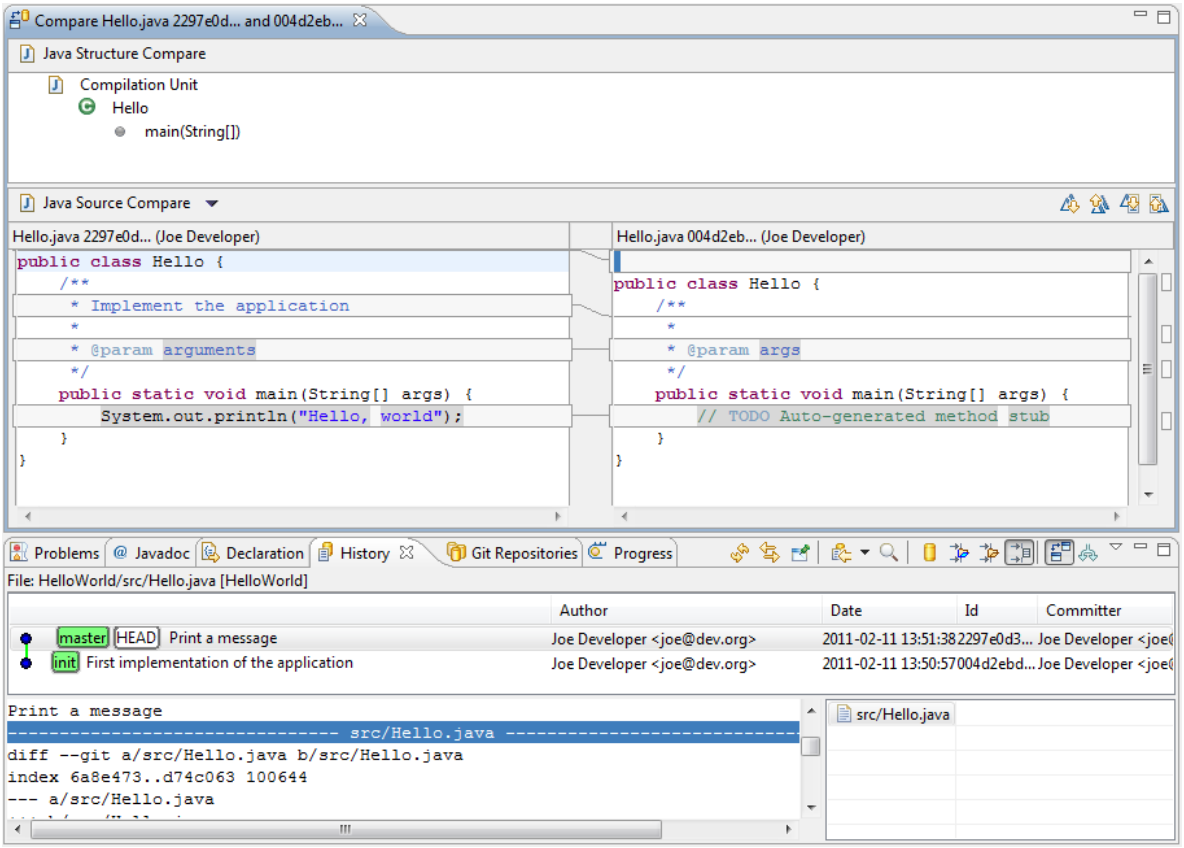



Abbildung 3.1: Verwendung der API CompareUI am Beispiel von eGit

### 3.2.3 Design der Bundles

In der Softwareentwicklung ist die Unterteilung der Software in kleinere Einheiten vorteilhaft, da dies den Test, die Wartung und die Entwicklung allgemein vereinfacht.

[1] Plug-in zur grafischen Bedienung der Versionierungssoftware Git

## 3.3 Implementierung

Dieser Abschnitt beschreibt die Implementierung der Komponenten für den Profilimport und die Aktualisierung der Filterprofile.

### 3.3.1 Entwicklungsumgebung

Die Implementierung der Bundles erfolgt in der IDE Eclipse[1]. Da RCP auf dem Eclipse-Framework basiert, liegt die Wahl dieser IDE nahe. Für das Konfigurationsmanagement wird das Build-Management-Tool Apache Maven 2 verwendet. Als Laufzeitumgebung wird die JRE[2]6 verwenden, da diese auch bei den bestehenden Bundles im Entwicklungsumfeld zum Einsatz kommt.

### 3.3.2 Komponente für den Profilimport

#### Extensions

Unter Verwendung des Extension-Point-Mechanismus von Eclipse RCP wurden in der Plugin.xml des Bundles verschiedene Extensions angelegt.

##### Grafische Steuerelemente

Für das ergänzen der Benutzeroberfläche der DBWB wird der Extension-Point `org.eclipse.ui.me` verwendet. Sowohl in der Toolbar der DBWB als auch in ihrer Menüleiste werden Buttons bzw Menüeinträge hinzugefügt. Das aktivieren eines solchen Menüeintrags oder Buttons startet den in Abschnitt 3.3.2 beschriebenen Handler. Listing 3.1 zeigt den Einsatz einer Extension, um eine neue Menükontribution anzulegen.

##### Handler

Neben den Extensions für die grafischen Steuerelemente wird eine Extension am Extension-Point `org.eclipse.ui.handlers` registriert. Dieser ImportHandler ist aktiv, falls in der Prozessperspektive ein Projekt selektiert ist, oder der Nutzer sich in der Dokumentenperspektive befindet (vgl. Abschnitt 3.2.2).

[1] Version Kepler
[2] Java Runtime Environment

Listing 3.1: Toolbar Extension

```
1
2  /**
3   * This is a doc comment.
4   */
5  // extension beispiel code
```

### Implementierte Klassen

#### ImportHandler

In der von `org.eclipse.core.commands.AbstractHandler` abgeleiteten Klasse ImportHandler wird zunächst überprüft, in welcher Perspektive sich der Benutzer beim Aufruf befand. Handelt es sich um die Prozessperspektive, wird mit Hilfe eines Selection-Services das selektierte Projekt im Projektnavigator ermittelt. Ein FileDialog Widget (`org.eclipse.swt.widgets.FileDialog`) wird geöffnet, über den der Benutzer das zu importierende Profil auswählen kann. Existiert kein Filterprofil des gleichen Typs im Projekt, so wird die Datei im profiles-Unterverzeichnis des Projekts kopiert und mit Hilfe des ProfileRepositoryLocationProvider im ProfileRepository der DBWB abgelegt. Existiert bereits ein Profil des gleichen Typs wird ein Dialog erzeugt, der dem Benutzer verschiedene Verfahrensmöglichkeiten bietet. Das im Projekt bestehende Profil kann überschrieben werden oder das zu importierende Profil kann umbenannt werden. Mit einem Enumerationsalgorithmus wird ein neuer Name erzeugt.

#### ProfileRepositoryLocationProvider

Der ProfileRepositoryLocationProvider wurde von einer firmeneigenen LocationProvider-Klasse abgeleitet. Dieser LocationProvider sucht nach, der DocBridge Mill Plus von Haus aus mitgelieferten, Filterprofilen und deren dazugehörigen XML-Schemata. Diese werden bereitgestellt und ins ProfileRepository kopiert. In der Klasse ProfileRepositoryLocationProvider werden viele Funktionen, die zur Ermittlung der Filterprofile dienen überschrieben, um eine eigene Profilversion zu erstellen. Ein Überladen des Konstruktors bietet die Möglichkeit, ein vom bestehenden System unabhängiges Profil einzubringen und ihm eine generierte Version zuzuordnen. Es wird eine eigene Location erstellt, mit der später das ProfileRepository aktualisiert werden kann.

#### Messages

### 3.3.3 Komponente zur Aktualisierung von Profilen

Da der Aufgabenzeitraum für die Analyse, Konzeption und Implementierung beider Komponenten zu knapp bemessen war, konnte die Aktualisierungskomponente nicht vollständig implementiert werden. Es existiert jedoch ein Prototyp, der mit der Technologie CompareUI(Abbildung 3.1), die bereits im Konzept erwähnt wird, realisiert wurde. Es wurde eine Standalone Eclipse RCP-Anwendung implementiert, die über die Funktionalität verfügt, einen Vergleichseditor mit zwei Profilen zu öffnen. Da bereits ein detailliertes Konzept besteht, dürfte die Implementierung dieser Komponente nicht allzu Zeitaufwendig sein. Vor allem, da Sie sich nicht in großem Maß von der Programmierung der Komponente für den Profilimport unterscheidet.

#### Verwendete Bibliotheken

## 3.4 Ergebnisse

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Listings

# Literaturverzeichnis

Ebert, R. (n.d.), *Eclipse RCP - Entwicklung von Desktop-Anwendungen mit der Eclipse Rich Client*. http://www.ralfebert.de/archive/eclipse_rcp/EclipseRCP.pdf.

Kriens, P. (2008), How osgi changed my life, Technical report. http://queue.acm.org/detail.cfm?id=1348594.

OSGiAlliance (2012), Osgi service platform specification, Technical report. http://www.osgi.org/Download/File?url=/download/r5/osgi.core-5.0.0.pdf.

# Abkürzungsverzeichnis

| | |
|---|---|
| **IDE** | Integrierte Entwicklungsumgebung |
| **RCP** | Rich Client Platform |
| **API** | Application Programming Interface |
| **OSGi** | Open Services Gateway initiative |
| **SWT** | Standart Widget Toolkit |
| **DBWB** | DockBridge Workbench |
| **GUI** | Graphical User Interface |
| **AFP** | Apple Filing Protocol |
| **PDF** | Portable Document Format |
| **XML** | Extensible Markup Language |
| **HTML** | Hypertext Markup Language |
| **JRE** | Java Runtime Environment |

# Glossar

### Mockup

Ein Mockup ist ein maßstäbliches Modell, das oft zu Präsentationszwecken oder zur Veranschaulichung genutzt wird.

### Toolbar

Eine Toolbar (Statusleiste) ist eine waagerechte oder senkrechte Leiste mit kleinen, häufig bebilderten Schaltflächen, die als erweiternde Elemente der Menüführung von Programmen mit grafischer Benutzeroberfläche dienen.