Hochschule Neubrandenburg
University of Applied Sciences

Fachbereich Landschaftsarchitektur, Geodäsie, Geoinformatik und Bauingenieurwesen

Studiengang
Geoinformatik

Bachelorarbeit
zum Thema

Anwendungsmöglichkeiten ausgewählter Rich Client Technologien am Beispiel des Überwachungs-, Auswertungs- und Analyseprogramm CIRCLE

Zum Erlangen des akademischen Grades
„Bachelor of Engineering“ (B.Eng.)

Vorgelegt von
Christoph Steckler

Erstprüfer: Prof. Dr.-Ing. Andreas Wehrenpfennig
Zweitprüfer: Dipl.-Ing. Michael Clausohm

Bearbeitungszeitraum: 27. Juli 2009 – 05. Oktober 2009

urn:nbn:de:gbv:519-thesis2009-0240-0

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Neustrelitz, den 05. Oktober 2009

Christoph Steckler

Danksagung

An dieser Stelle möchte ich allen Personen danken, die mich bei der Erstellung der Abschlussarbeit unterstützt haben.

Ein besonderer Dank gilt meinen Betreuern Herr Prof. A. Wehrenpfennig und Herr Dipl.-Ing. M. Clausohm für ihre Hilfe und Ratschläge.

Ein besonderer Dank geht an meine Familie, die mich während des Studiums unterstützt haben und immer an mich glaubten.

An meine Freundin richtet sich ein außerordentlicher Dank. Sie hat mich immer unterstützt, und an mich geglaubt. Marie-Christin hat mich in schwierigen Phasen wieder aufgebaut, mich motiviert und mir jedes Mal Mut zugesprochen.

Kurzfassung

Im Zuge der Bachelorarbeit soll eine von fünf ausgewählten Rich Client beziehungsweise Rich Internet Application Technologien gewählt werden, die für die Erweiterung des Überwachungs-, Auswertungs- und Analyseprogramm CIRCLE dienen. Zu Beginn erfolgt eine Einführung in die Anwendung. Anschließend werden die ausgewählten Rich Client und Rich Internet Application Technologien vorgestellt und verglichen. Bei dieser Auswahl handelt es sich um Adobe Flex/AIR, Microsoft Silverlight, JavaFX, Rich Ajax Platform (RAP) und die EclipseRCP (Java). Anhand gegebener Kriterien wird ein Kandidat favorisiert und ein Konzept für den neuen CIRCLE Client angefertigt. Ferner werden weitere Softwarestandards respektive Frameworks wie OSGi und der Nachrichtendienst ActiveMQ in die Arbeit einbezogen, da im Laufe der Bachelorarbeit ebenfalls Änderungen am CIRCLE System vorgenommen werden. ActiveMQ und OSGi übernehmen somit auch im neuen Rich Client einen wichtigen Teil.

Abstract

In the Course of this Bachelor one of five Rich Client or respectively Rich Internet Applications is chosen to upgrade the monitoring, evaluation an analysis system CIRCLE. In the beginning there is an Introduction of the application followed by introducing and comparing the selected Rich Clients and Rich Internet Application technologies. The Selection is made up of Adobe Flex/AIR, Microsoft Silverlight, JavaFX, Rich Ajax Platform and EclipseRCP. Within given criteria one technology is favored and a concept of the new CIRCLE Client is created. Additionally new techniques like the OSGi Standard and the Message Broker application ActiveMQ are included in the thesis because of ongoing changes in the CIRCLE system.

Inhaltsverzeichnis

1. Einleitung

Will man in heutiger Zeit Anwendungen programmieren, steht man nicht nur vor der Wahl ob es eine Webapplikation oder eine Desktopanwendung werden soll, man hat auch eine Vielzahl von Hilfsmitteln. Dazu zählen zum Beispiel auch die Frameworks, wovon es ebenfalls eine große Anzahl gibt. Sie warten mit optisch ansprechenden Möglichkeiten zur Gestaltung der Benutzeroberfläche auf. Die Aktualität dieser Technologien führte auch zum Entstehen dieser Bachelorarbeit. Die Firma Clausohm möchte diese Möglichkeiten der Rich Client und Rich Internet Application Technologien nutzen.

Ziel dieser Bachelorarbeit ist das Prüfen und Anwenden von Rich Client und Rich Internet Technologien für das Steuerungs-, Überwachungs- und Analyseprogramm CIRCLE der Firma Clausohm GmbH. In diesem Zuge wird die CIRCLE Struktur erneuert. Aufgabe ist es ein neues Konzept für den CIRCLE Client zu entwerfen und die Programmoberfläche anzupassen. Zur Erfüllung dieser Aufgabe werden mehrere Lösungsansätze im Bereich der Rich Client und Rich Internet Application begutachtet. Der Schwerpunkt liegt jedoch auf der Programmiersprache Java unter Verwendung einer Implementierung des OSGi Standards durch die IDE Eclipse. Aufbauend darauf wird geprüft, welche weiteren Rich Clients zur Gestaltung der GUI[1] kombiniert werden können. Ausgehend von jenen Ergebnissen wird eine Technologie favorisiert und ein Prototyp erstellt.

Im Anfangsstadium der Arbeit war eine Suche und Auswahl einer Rich Client- beziehungsweise Rich Internet Application Technologie für CIRCLE vorrangig. Während des Voranschreitens der Abschlussarbeit hat sich das Thema etwas verschoben. Als neue Optionen für den Rich Client und das CIRCLE System sind die OSGi Spezifikation und der Message Broker ActiveMQ eingeführt worden und finden in dieser Arbeit Beachtung. Sie gliedert sich in drei Kernteile. Zu Beginn wird CIRCLE in Auszügen vorgestellt, der Fokus liegt auf dem Analyzer und der GUI. Das darauf folgende Kapitel befasst sich mit den verschiedenen Rich Client und Rich Internet Technologien. Es werden einzelne Technologien

[1] Graphical User Interface – stellt die Grafische Nutzerschnittstelle mit der Anwendung dar

ausgewählt, die unterschiedlichen Ansätze begutachtet und verglichen, sowie der Nutzen für CIRLCE betrachtet. Aufbauend auf den Ergebnissen aus Kapitel 3 wird im vierten Abschnitt ein Konzept für einen neuen CIRCLE Analyzer erstellt.

Die Firma, in der die Bachelorarbeit anfertigt wurde, ist die Clausohm Software GmbH. Sie wurde 1990 als Softwarebüro gegründet und 1994 in eine GmbH umgewandelt. Die Firma ist in mehreren Tätigkeitsfeldern vertreten. Das sind zum einen die Erstellung von Komplettlösungen im Umfeld des Maschinenbaus, zum anderen die Webprogrammierung, sowie die Fertigung von Elektroausrüstung von Maschinen. Ersteres beinhaltet die Softwareerstellung, die Konfiguration der Hardware, sowie deren Auslieferung und Installation. Mit Ihren Lösungen ist die Clausohm Software GmbH national und international tätig. Beschäftigt werden 43 Mitarbeiter, darunter Ingenieure, Konstrukteure, SPS-Programmierer, sowie Elektriker.

2. Analyse

Dieser Abschnitt widmet sich dem bestehenden Programm CIRCLE. Es werden die Aufgaben, die technischen Daten sowie der Aufbau von CIRCLE begutachtet. Im Speziellen wird auf den CIRCLE Analyzer eingegangen, welches das Visualisierungsprogramm des ganzen Systems darstellt. Da im Zuge der Bachelorarbeit mit Hilfe von Rich Client Technologien die Möglichkeiten einer neuen Oberfläche für das CIRCLE System geprüft werden sollen, ist eine Betrachtung der momentanen Ausgangslage äußerst wichtig. In diesem Kapitel soll die Aufgabe geklärt und abschließend die Anforderungen an die Rich Client Technologien angesprochen werden.

2.1. Aufgabe von CIRCLE

Die Bezeichnung CIRCLE ist ein Akronym und setzt sich aus folgenden Einzelteilen zusammen: Control-Information-Recording-Communication-Labeling-Evaluation. Diese sieben Begriffe beschreiben die Aufgabenfelder der Anwendung. Es bietet die Kontrolle über die Fertigungsanlagen. Die von der Clausohm Software GmbH entwickelte Unternehmensanwendung ist auf den Bereich der maschinellen Produktion von Gütern abgestimmt. Als Teil eines ganzen Systems übernimmt CIRCLE die Auswertung, Überwachung und Anzeige von Produktionsprozessen anhand der aufgenommenen Produkt-, Prozess- und Maschinendaten. Maschinen in diesem Zusammenhang sind als Fertigungsanlagen für bestimmte Produkte zu verstehen, wie zum Beispiel verschiedenste Leuchtmittel. Es ist nicht nur ein Informationssystem, CIRCLE dient ebenfalls zur Regelung der Maschinen und zur Sicherung der

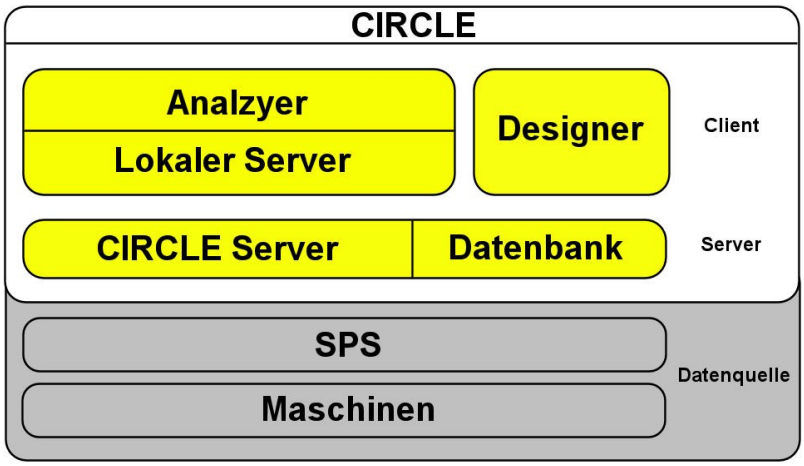

Abbildung 1 – Grundlegende CIRCLE Komponenten

Produktqualität. Es bietet eine Fülle von Statistikmöglichkeiten, angefangen bei der Anzeige der Anzahl an produzierten Gütern über Prozessstatistiken bis hin zu Effizienzstatistiken der produzierenden Maschinen. Mit Hilfe der aufgenommenen Daten kann eine Systemoptimierung vorgenommen, Produktionsstätten effektiver genutzt und eine gleichbleibende oder gesteigerte Qualität der Produkte erreicht werden.
Zusammenfassend besitzt CIRCLE zwei große Anwendungsgebiete in einem Unternehmen, das sind der Produktionsbereich und das Management. Die mit CIRCLE erhobenen Daten können für unterschiedliche Bereiche hilfreich sein. Durch die Erfassung der Maschinendaten ist es dem Personal möglich eine Leistungsverbesserung vorzunehmen, was sich positiv auf die Qualität und deren Sicherung auswirkt. Ferner hilft es dem Management bei der Planung der Produkte.

2.2. Momentane Ausgangslage

CIRCLE ist eine reine Softwareanwendung und kann in mehrere Komponenten geteilt werden. Darunter fallen der CIRCLE Client und Server mit der Datenbank, sowie der CIRCLE Analyzer. Jedoch fehlt noch die SPS. Dieses Kürzel steht für Speicherprogrammierbare Steuerung. Oft wird diese auch als PLC bezeichnet, der englischen Version, dem Programable Logical Controller. Ohne diese Hardware wäre die Anwendung nicht voll funktionsfähig, da die Produktionsdaten fehlen würden. Die SPS sind elektronische Baugruppen, die zur Steuerung und Regelung von Maschinen eingesetzt werden. Sie besitzen eine Firmware und maschinenspezifische Anwendungen, womit eine Steuerung und Datenübermittlung gewährleistet wird. Je nach Bedarf werden Elemente kombiniert. Eine Verbindung erfolgt in heutigen Systemen vorrangig über einen Datenbus. Über Sensoren an der Maschine werden Daten gelesen, Stellglieder ermöglichen die Umsetzung der Steuerungsbefehle.
Die SPS Schränke werden in der Firma zusammengebaut und dann ausgeliefert. Es können aber auch bereits installierte Varianten genutzt werden. Aufgrund standardisierter Protokolle in der SPS Steuerung spielt dieser Aspekt für CIRCLE jedoch keine Rolle.

Fertigungsanlagen bestehen in der Regel aus mehreren Maschinen. Diese Tatsache wird durch mehrere SPS berücksichtigt, die Daten jeder Maschine liefern. CIRCLE ist in der Lage mehrere SPS zu verwalten und diese Daten gruppiert sowie auch einzeln anzuzeigen und auszuwerten.
Circle wird in der jeweiligen Produktionsanlage eines jeden Kunden auf den dafür vorgesehenen Computern direkt vor Ort eingerichtet. Durch ein entsprechendes Tool ist eine Fernwartung der Software, Fehlerbehebungen, Verbesserungen und das Aufspielen neuer Versionen gewährleistet.

2.2.1. Tech. Daten - Aufbau

CIRCLE befindet sich seit 1994 in Entwicklung. Ursprünglich wurde es im Auftrag der Philips Lighting GmbH Aachen entwickelt. Somit wird es in verschiedenen Philips Fertigungsanlagen genutzt. Die Programmiergrundlage ist die Programmiersprache C/C++. CIRCLE wurde vollständig damit erstellt. Aus historisch gewachsenen Gründen werden zum Teil ältere Compiler genutzt, weshalb nur Windows NT, Windows 2000 und Windows XP unterstützt werden.
Wie bereits erwähnt nutzt CIRCLE Datenbanken um die anfallenden Daten zu speichern. Hierfür wird eine Lösung der Firma Birdstep eingesetzt. Der Birdstep RDM Server ist eine in die Anwendung integrierte Datenbank, also eine sogenannte embedded Database. Der RDM Server ist auf C/C++ Anwendungen optimiert. Sie verwendet Techniken der relationalen und der Netzwerk-Datenmodellierung. Sie besitzt ein proprietäres Speicher- und Abfrageformat. Es werden aber auch SQL Queries unterstützt. Die Datenbank befindet sich beim Client-Server System auf dem Datenbankserver im Stand-Alone System auf dem Client PC. Die Datenbanklösung ist für den Datenaustausch auf schnelle, zeitnahe Prozesse ausgerichtet. So werden in CIRCLE bis zu 30 Telegramme pro Sekunde gesendet, eine Verarbeitung erfolgt in wenigen Millisekunden. Eines dieser Telegramme kann bis zu 16KB groß sein, so dass sich ein Datendurchsatz von 480Kb/s ergibt.
Um einen zentralen Zugriff auf Daten zu erhalten, wird ein Datenbankserver eingerichtet. Von diesem können alle Clients ihre Informationen beziehen. Dabei Fallen unterschiedlichste Daten an. Diese können kategorisiert werden. Sie werden produktorientiert gespeichert, das heißt die Daten werden für jedes

einzelne Produkt, welches durch die verschiedenen Maschinen gefertigt wurde, zusammenhängend gespeichert. Alle Daten einer einzelnen Maschine stellen einen Produktsatz dar. Zu einem vollständigen Produkt gehören mehrere Fertigungsschritte und damit auch mehrere Maschinen. Die Daten werden in diesem Fall mit einem Label versehen, einer Art Verknüpfung von zusammenhängen Daten. Diese können dann speziell für ein Produkt ausgelesen werden. Telegramme, die Produktdaten übermitteln, werden pro Maschinentakt gesendet.

Die Einteilung der Daten wurde in CIRCLE folgendermaßen vorgenommen,

Die Stammdaten bestehen aus:

- Gruppeninformationen
- Anzahl der Maschinen
- Anzahl der Platzhalter auf einer Maschine
- Ausfall-/Leerlauftypen
- Messwertdefinitionen
- Maschinenereignisse(welche Möglichkeiten es auf Maschinen geben kann)
- statische Texte
- Beschreibung der Produkttypen
- Beschreibung der benötigten Materialien und Halbfabrikate
- Materialfluss für jeden Produkttyp

Fertigungsdaten:

- Produktdaten
- Statistikdaten
- statistische Prozessdaten
- Ereignisdaten/Maschinendaten
- Materialdaten
- Produkttypen und Schichten
- Prozessgrenzen

Wie man eventuell erkennen kann, sind Fertigungsdaten fortlaufende dynamische Informationen, die während des Betriebs erhoben werden. Eine weitere Aufschlüsselung der Daten ist möglich, jedoch für den Zweck der Arbeit von geringerem Interesse.

Die Eingabe und Wartung der Stammdaten erfolgt im CIRCLE Designer. Der Designer wird vor allem in der Planungsphase eingesetzt. So müssen die vielfältigen Produkt-, Prozess- und Maschinenparameter bestimmt und umgesetzt werden. Die Einstellungen gewähren die korrekte Funktionsweise von CIRCLE. Fehleinstellungen resultieren in falsch ausgewerteten Daten.

2.2.2. Kommunikationsablauf

CIRCLE besitzt eine Besonderheit. Der Analyzer benötigt einen gestarteten Server. Dieser ist auf dem Client und auf dem Server die gleiche Anwendung. Es beherrscht, wenn als Server konfiguriert, die Abfrage der Daten aus der Datenbank, die Kommunikation mit den SPS und dem Client. Im Client-Modus ist der Server für den Datenaustausch zwischen zentralem Server und Analyzer zuständig. Durch diese Funktionsweise kann CIRCLE als Stand-Alone System, aber auch als verteiltes System in einem Netzwerk fungieren. Ist es in einem Netzwerk realisiert, sind Client und Server über die Ethernet Schnittstelle miteinander verbunden, ebenso wie die SPS und der Server. Ältere Versionen nutzen eine 2Port Karte für die Kommunikation zwischen Server und SPS. Der zentrale Server übernimmt die Kommunikation mit allen SPS, jeder Client kann auf den Server zugreifen, wobei dieser auf einem anderen System installiert ist. Als Stand-Alone Variante arbeiten Client und Server auf einem PC.

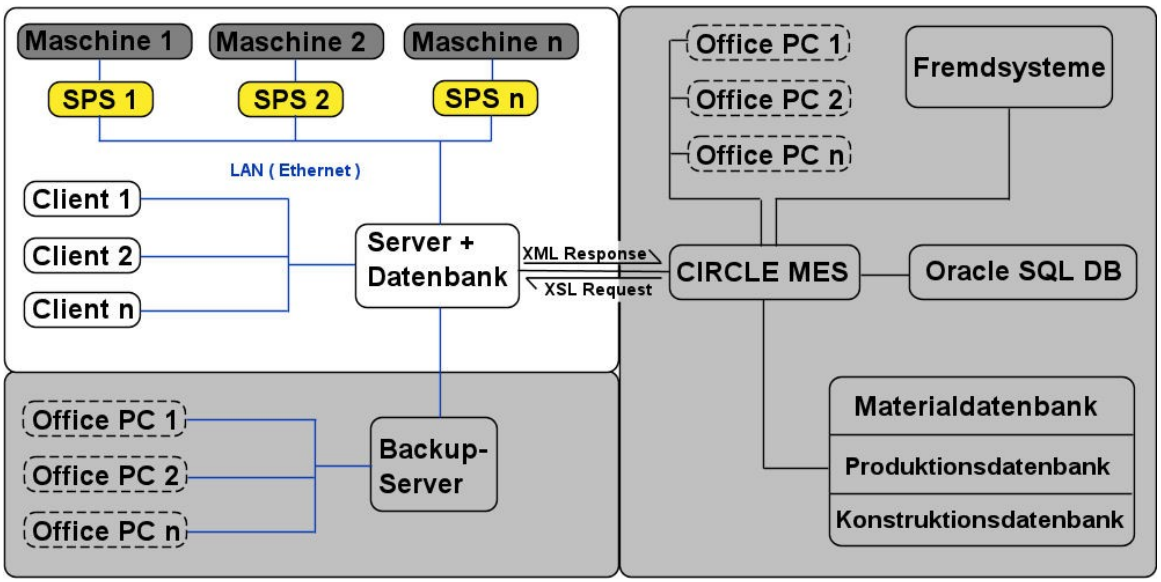

Abbildung 2 – CIRCLE System - Kommunikationsstruktur

Im gesamten Kontext sieht die Kommunikationsstruktur des CIRCLE Systems wie auf dem oben dargestellten Bild aus. Der weiß hinterlegte Teil ist für Bachelorarbeit von Bedeutung. Die restlichen Komponenten dienen dem Gesamtüberblick über CIRCLE. CIRCLE MES ist eine Webanwendung. Sie ist speziell auf das Management einer Firma angepasst. Über dessen Services können ERP System, wie zum Beispiel von SAP die Daten empfangen. Der Backupserver dient, wie oben abgebildet, der Archivierung der Daten, sowie auch dem Backup der Daten. Im Fall dieser Abbildung kann der Backupserver auch als Quelle der Office-Auswertungen dienen. Circle Anwendungen werden auf diesem Server nicht installiert.

Will ein Nutzer aktuelle Daten angezeigt bekommen kann man sich folgenden Ablauf vorstellen. In folgender Grafik ist der allgemeine Ablauf von einer Eingabe bis zu Ausgabe dargestellt.

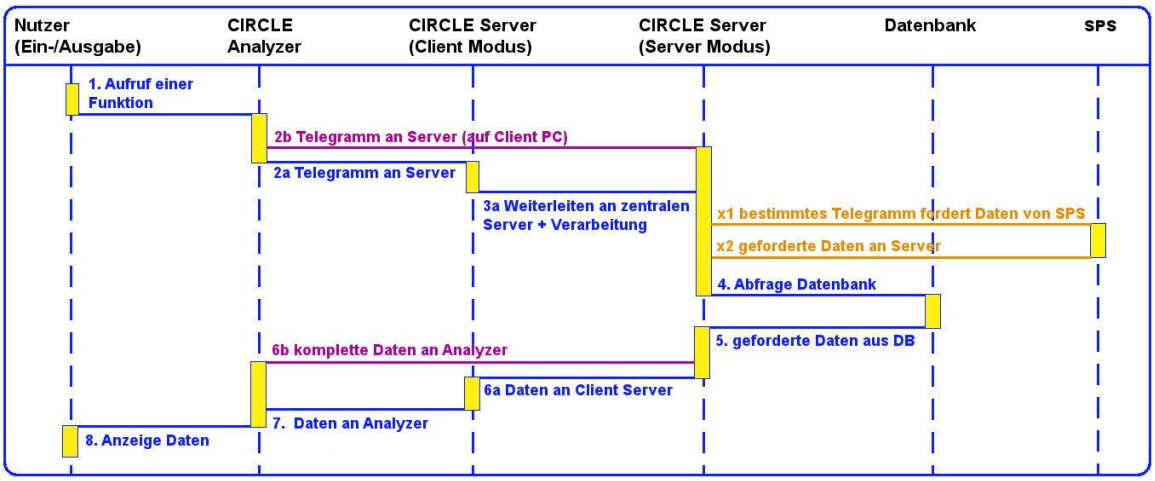

Abbildung 3 – Sequenzdiagramm

Die in Blau gehalten Abschnitte stellen die normalen Aktivitäten im verteilten System dar. Die Magenta gefärbten Stränge bilden den Unterschied zwischen dem Stand-Alone System und dem verteilten System. Die Orange-farbenen Elemente sind optionale Zugriffe auf die SPS, welche durch die Art der übermittelten Telegramme bestimmt werden.

Es wird von einem bereits gestarteten Client und Server ausgegangen. Der Anwender macht eine Eingabe, in diesem Fall ist es die Auswahl einer Funktion, die genaue Angabe ist für das Schema nebensächlich(1). Durch die Eingabe werden verschiedene Prozesse im CIRCLE Analyzer ausgelöst. So sendet der CIRCLE Analyzer ein Telegramm an den lokalen Server(2a). Dieses Telegramm wird vom Analyzer erstellt und enthält Parameter, mit welchen der weitere Kommunikationsprozess gesteuert wird. Beide Komponenten kommunizieren über Named Pipes, ein Übertragungsprotokoll. Es ist ein gepufferter Datenstrom nach dem First In – First Out Prinzip. Die Daten, die von einem Prozess ausgegeben werden, dienen einem anderen Prozess bzw. Anwendung als Eingabe. Der Server im Client Modus schickt die Anweisungen an den Server(3a). Auf diesem befindet sich die Servervariante des Datenbankservers. Dieser hat die Aufgabe die anfallenden Produktions- und Maschinendaten der SPS zu verwalten und in einer Datenbank abzulegen. Der Server verarbeitet das Telegramm entsprechend den darin enthaltenen Anweisungen. Einige

Telegrammtypen enthalten bestimmte Anweisungen, die es erforderlich machen, Daten separat von der SPS zu erfragen(x1+x2). Vorrangig sind dies maschinenbezogene Daten. Nachdem die Datenbankabfrage erfolgt(4) und ein Datensatz geliefert wird(5), werden die Daten zusammen mit den Informationen aus der SPS an den Server auf dem Client gesendet(6). Um Datenaufkommen zwischen dem Server und dem Analyzer zu verringern, werden ankommende Daten(7) durch die CIRCLE Anwendung in Pools geschrieben. Diese Pools sind als Memory Mapped Files angelegt. Der Server im Client Modus besitzt nur den schreibenden Zugriff auf die Mapfiles, während der Analyzer nur lesende Rechte erhält. Die gelesenen Daten werden dann im Analyzer dargestellt(8).

Wird ein Stand-Alone System genutzt, verhält sich der Datenaustausch ähnlich. Der wichtige Unterschied ist jedoch der fehlende zentrale CIRCLE Server. So übernimmt der lokale Server alle Aufgaben, die sonst auf dem zentralen Datenbankserver laufen. Die SPS ist direkt mit dem CIRCLE Client verbunden. So werden Telegramme vom Analyzer an den auf dem Client installierten Server geschickt(2b). Dieser befindet sich im Server Modus. Die Abfrage der Daten bleibt gleich. Analog zur Datenabfrage erfolgt der Datenempfang(6b), sowie der restliche Ablauf.

Unabhängig zum beschriebenen Kommunikationsablauf verläuft die Abfrage und Speicherung der Daten der SPS. Die Daten werden in regelmäßigen Abständen von der SPS an den Server gesendet und dort in die Datenbank geschrieben. Produktionsdaten werden automatisch an den Server übermittelt. Maschinendaten hingegen, wie etwa der aktuelle Zustand, werden per Abfrage vom Server erfasst. Sind die Clients nicht verfügbar, werden die anfallenden Daten natürlich weiterhin gespeichert. Besteht der Fall, dass die Verbindung zum Server unterbrochen wird, so werden die Daten im Speicher der SPS gehalten. Die Datenmenge ist abhängig von der verbauten Menge an Speicher.

2.3. CIRCLE Analyzer

Auch wenn CIRCLE aus mehr als nur dem CIRCLE Analyzer besteht, wird dieser Anwendung mehr Aufmerksamkeit gewidmet, da im Rahmen der Bachelorarbeit ein Rich Client des CIRCLE Analyzer entstehen soll. Seit der Erstellung von CIRCLE wurde es ständig erweitert, jedoch blieb die grafische Oberfläche weitgehend unangetastet. So besitzen die Grafikelemente ein typisches Windows 98/2000 Aussehen. Mit den Jahren haben sich die Anforderungen an die Bedienung und das Aussehen der Programmoberfläche geändert. Mit Hilfe neuer Lösungen soll ein aktueller Stand erreicht werden.

Der CIRCLE Analyzer ist das grafische Anzeige- und Auswertungsprogramm des ganzen Systems. Es stellt die Schnittstelle zwischen Mensch und PC dar. Es wird auf jedem Client PC als Desktopanwendung installiert.

Der Analyzer allein ist nicht funktionsfähig, es wird immer der CIRCLE Server benötigt. Dabei ist es egal ob der Server Daten empfängt oder nicht. Wie bereits erwähnt findet im Analyzer die Auswertung der Produktions- und Maschinendaten statt. Dafür stehen umfangreiche Funktionalitäten bereit.

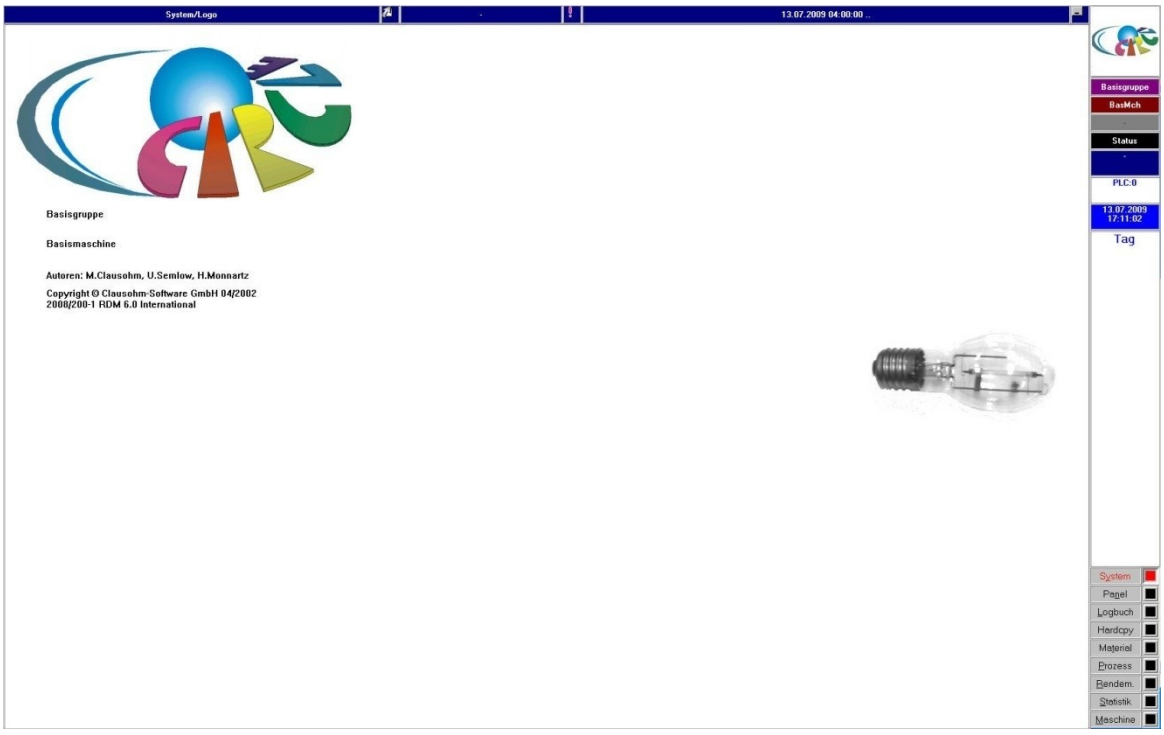

Abbildung 4 – CIRCLE Startbild

Das Bild zeigt den Startbildschirm des Analyzer. Die Anwendung ist in drei Bereiche gegliedert, welche im weiteren Ablauf näher betrachtet werden. Den größten Bereich nimmt das Prozessfenster ein. In diesem wird die zuletzt aufgerufene Funktion angezeigt. Die Leiste auf der rechten Seite wird zur Bedienung der Anwendung verwendet und ruft die Funktionen auf. Weitere Informationen erfolgen in der Betrachtung der einzelnen Bereiche des CIRCLE Analyzer. Die obere Leiste bietet weitere Bedienelemente. Werden bestimmte Elemente aufgerufen, stehen neue Bedienmöglichkeiten zur Verfügung. Jeder Menüpunkt in der Seitenleiste, bietet weitere Untermenüs, ähnlich dem Windows Startmenü.

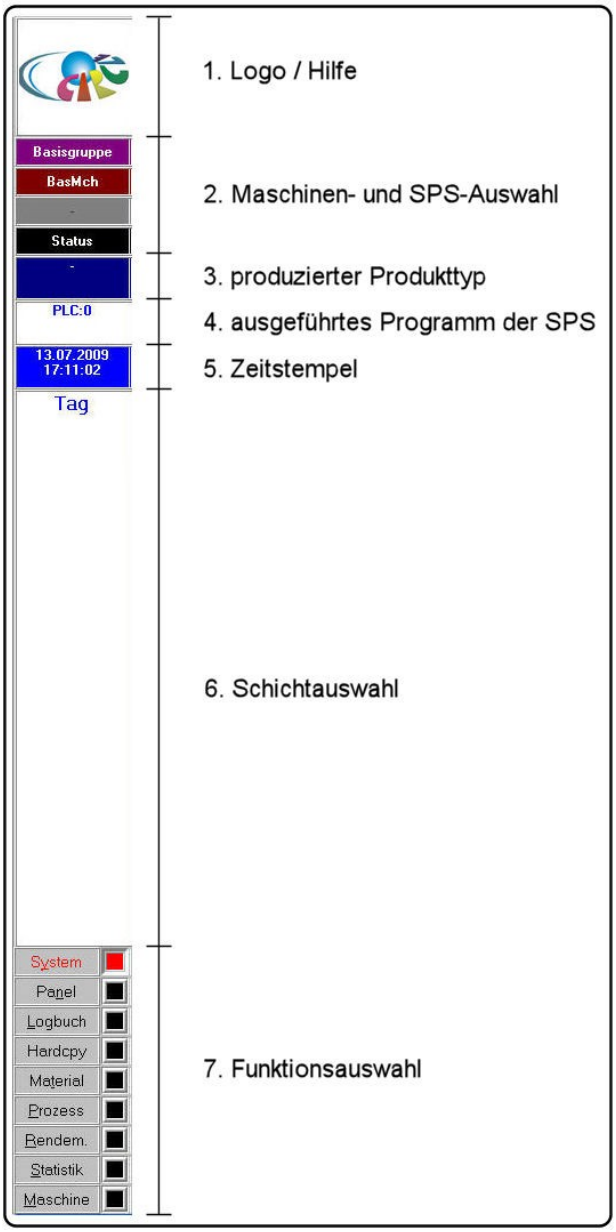

Abbildung 5 – CIRCLE Analyzer Seitenleiste

Die Seitenleiste besteht ebenfalls aus mehreren Unterteilungen. Sie enthält die wichtigsten Auswahlmöglichkeiten. Oberstes Feld zeigt das entsprechende Firmenlogo und ruft die Hilfe auf(1). Die nächsten vier Felder werden für die Auswahl der Maschinen genutzt(2). Das Obere davon wird zur Auswahl der Maschinengruppe genutzt. Das darunterliegende Feld ermöglicht die Selektion einer Maschine innerhalb der Maschinengruppe gefolgt von der Location, was eine Untergruppe einer Maschine mit eigenen Haltern[2] ist. Der letzte Abschnitt zeigt den Status der gewählten Maschine. Darunter befindet sich die Anzeige des produzierten Produkttyps(3). Im nächsten Feld wird das Programm der SPS bestimmt(4), gefolgt vom Zeitstempel(5) und der Schichtauswahl(6). Den letzten Abschnitt bilden die Funktionsknöpfe, welche wiederum Untermenüs aufrufen(7). Um einen Überblick über die in CIRCLE verwendeten Funktionen zu geben zeigt die nachfolgende Tabelle die volle Funktionsauswahl der Seitenleiste. Eine Aufschlüsselung der einzelnen Funktionen ist nicht vorgesehen. Es soll nur der Umfang aufgezeigt werden.

System	Ende
	Logo
	Export
	Roh Export
	Dokumente
	Skript
	Designer
Panel	Halter
	Messwerte
	Vorgaben
	Zeitplaner
	Aufgaben
	Störung
Logbuch	Anzeige
	Laufzeiten
	Pareto
	Ereignisse
	Produkttypen
	Schichtzeiten
	Notizen
	Grenzen
	Aufgaben

Prozess	Aktuell
	Direkt
	Statistik
	UniKorr
	Histogramm
	Prozessliste
	Mittelwert
	Analyse
	Gleitwertkarte
Rendement	Gesamtstatistik
	Laufzeitstatistik
	Produktstatistik
	Zeitdiagramm
Statistik	Stoppstatistik
	Ausfallstatistik
	Leerlaufstatistik
	Halter
	Ereignis
Maschine	Aktuell
	Produktstatistik
	Proben

[2] Halter sind einzelne Stationen, die ein Produkt innerhalb einer Maschine durchläuft

Hardcopy	-	Visuell
Material	Input	Stichproben
	Output	Wartung
		Wartung
	Lager	Statistik
	Logbuch	Wartung Log
	Batch	

Tabelle 1 – Funktionsumfang der Seitenleiste [1]

Abbildung 6 – Kopfzeile des CIRCLE Analyzers

Der letzte betrachtete Bereich ist die Kopfleiste. Sie enthält die Funktionen Drucken, XML Export und die Aufgabenlisten und sollten selbsterklärend sein. Eine weitere Funktion ist die Anzeige des Anzeigezeitraums, welche über einen Kalender bestimmt wird. Dieser wird über ein Klicken des angezeigten Datums in der Kopfzeile aktiviert. Der Anzeigezeitraum hat auf viele Statistiken Auswirkungen. Es werden Daten entsprechend des angegebenen Zeitraums angezeigt. Abschließend soll das Funktionsfenster betrachtet werden. Es nimmt den größten Bereich der Anwendung ein. Dort werden alle Funktionen abgebildet. Es ist stets nur möglich eine aufgerufene Funktion anzuzeigen. Je nach gewählter Option wird das Drucksymbol in der Kopfzeile angezeigt. Somit wurde ein Überblick über den CIRCLE Analyzer gezeigt.

2.4. Pläne für die Erneuerung

Der aktuelle Stand wurde soeben beleuchtet, nun zum künftigen Ziel. Vorerst wird aber geklärt warum überhaupt ein Technologiewechsel anstrebt wird. Einige Gründe wurden bereits im Vorfeld genannt, sollen hier aber noch einmal zusammengetragen werden. Ein wichtiger Grund für den Wechsel besteht in der Inkompatibilität mit dem C++ Compiler und dem neusten Windows Betriebssystem und den somit resultierenden Problemen des CIRCLE Analyzers. Hier möchte man aktuellere Technologien nutzen. Ein weiterer Grund ergibt sich durch den Programmaufbau. In den vergangenen Jahren wurde CIRCLE den entsprechenden Bedingungen der Kunden angepasst und erweitert. Das

Programm hat sich teilweise, auf Grund der schrittweisen Erweiterungen, zu einem monolithischen System entwickelt. Eine Wartung der Softwarekomponenten wird dadurch aufwändig, da Prozesse ineinander greifen und man für eventuell kleine Änderungen viel Programmierarbeit aufgebracht werden muss.

Durch die Auswahl einer Rich Client Technologie beziehungsweise einer Rich Internet Application sollen oben genannten Inkompatibilitäten gelöst werden, da die Entwicklertools gewechselt werden. Diese ermöglichen die Nutzung des neuen CIRCLE Clients auf aktuellen Betriebssystemen. Der neue CIRCLE Client soll einem monolithischen System entgegenwirken und modularer konzipiert werden. Als Lösung wurde die OSGi Spezifikation gewählt, welche das Konzept der Modularisierung berücksichtigt und weitere Vorteile zu bieten hat. Auf das Framework wird im weiteren Verlauf der Arbeit eingegangen. Ziel der Arbeit ist es jedoch unter verschiedenen Rich Client Technologien eine Möglichkeit zu wählen, um dem CIRCLE Analyzer eine neue Oberfläche zu geben.

Das bestehende CIRCLE wird auch weiterhin genutzt. Es erfolgt also eine schrittweise Einführung ausgewählter Komponenten. Die neue Anwendung hat noch keinen finalen Namen, somit wird im weiteren Verlauf Namen wie CIRCLE RC oder CIRCLE Rich Client verwendet. In jedem Fall ist das gleiche gemeint. Der CIRCLE Rich Client ist für neue Aufgaben und Projektbereiche vorgesehen. Er wird keine 1:1 Portierung des CIRCLE Analyzers. Einige Funktionen werden nicht übernommen, dafür sollen Neue eingebaut werden. Vor allem soll der CIRCLE RC von den aktuellen Technologien profitieren. Das vorläufige Ziel ist die Erweiterung des CIRCLE Systems, so dass alle Bestandteile genutzt werden. Zur Bewältigung dieser Aufgabe muss ein System verwendet werden, welches die Kommunikation zwischen den Bereichen des CIRCLE Systems steuert. Demzufolge wird ein asynchrones Message System eingeführt, welches die Datenübertragung regelt. Weitere Details dazu werden in einem eigenen Abschnitt im Laufe der Arbeit begutachtet. Wie man bemerkt, werden nicht nur die Oberfläche von CIRCLE, sondern weitere wesentliche Strukturen von CIRCLE, erneuert. Zusammengefasst sind es die grafische Oberfläche, der Programmaufbau und die Kommunikation zwischen verschiedenen Schnittstellen.

2.4.1. Anforderungen

Neben der Bedingung einer aktuellen Darstellung der Oberfläche durch einen Rich Client, des Modularen Programmaufbaus und dem Datenaustausch sind weitere Bedingungen an den Umstieg geknüpft. So sollen nach Möglichkeit folgende Forderungen berücksichtigt werden.

- möglichst plattformunabhängig
- Im Webbrowser und auf Desktop nutzbar
- Zukunftssichere Technologie
- Hot Deployment[3]
- Verwendung eines eigenen Designs/eigener Grafikelemente
- Geringerer Wartungsaufwand
- Funktionen erweitern, Auswertung aufbereiten
- Mehr Komfort bei der Erstellung von Projekten

Wie man an der Auflistung erkennt, sind einige Punkte sehr allgemein gehalten und werden von vielen verschiedenen Softwareentwicklern verfolgt. Mit Hilfe ausgewählter Technologien wird versucht diesen Anforderungen gerecht zu werden. Mit Hilfe der Rich Clients soll eine weitestgehende Betriebssystemunabhängigkeit erreicht werden. Dadurch sind die Kunden nicht mehr nur an Windows gebunden und können somit flexibler arbeiten. Ein weiterer Schritt zur flexibleren Arbeit ist die Nutzung der Anwendung sowohl auf dem Desktop als auch in einem Webbrowser. Die verwendeten Technologien sollen dabei auch zukunftssicher sein, sollen also die nächsten Jahre Bestand haben, da man die Anwendungen nichtfortlaufend neu programmieren möchte. Viele Unternehmen haben meist nicht die finanziellen Spielräume oft auf neue Technologien umzuschwenken. Mit dem Umbau des CIRCLE Systems erhofft man sich auch Erleichterung bei der Wartung und Pflege von CIRCLE, damit man sich auf Kernelemente wie die Weiterentwicklung konzentrieren kann. Die Möglichkeiten der Rich Client Technologien sollen nicht nur auf die reine Oberfläche angewendet werden, mit deren Hilfe sollen Funktionen erweitert und aufgewertet werden.

[3] Updates während der Ausführung der Anwendung einspielen ohne einen Neustart zu benötigen

3. Möglichkeiten der Rich Client / Rich Internet Application

Nachdem CIRCLE vorgestellt wurde und die Anforderungen an eine Rich Client Version des CIRCLE Analyzers geklärt wurden, wird nun auf die ausgewählten Frameworks eingegangen. Zur Wahl stehen JavaFX, Microsoft Silverlight, Adobe Flex und AIR und das RAP Framework. Zusätzlich wird Java mit der Eclipse RCP Entwicklungsumgebung hinzugezogen und mit den anderen Technologien verglichen. Es gibt natürlich noch weitere Frameworks, doch um die Arbeit einzugrenzen, beschränkt sich die Auswahl auf fünf Kandidaten. Alle aufgezählten Frameworks genießen eine gewisse Aktualität und Popularität, weshalb sie ausgewählt wurden. JavaFX steht als jüngster Zuwachs in der Reihe der Rich Internet Application in Konkurrenz mit dem weitverbreiteten Adobe Flash und Microsoft Silverlight. Die Eclipse Rich Client Platform steht im Kontext für traditionelle Java Anwendungen, die durch einige Eclipse Techniken erweitert werden. Mit der Rich Ajax Platform lassen ohne weitere Plug-Ins, nur mit Hilfe von JavaScript und XML, Desktop- und Webanwendungen erstellen.
Nun fällt auf, dass auch einige als Rich Internet Application, im weiteren Text auch als RIA bezeichnet, bekannte Technologien aufgezählt werden. Das hat seine Richtigkeit, denn es ist nicht festgelegt ob der neue Analyzer als Desktop- oder als Webanwendung betrieben werden soll. Somit wird geprüft ob und in wie weit es möglich ist, die genannte Anforderung zu realisieren. Im Idealfall soll der Rich Client für CIRCLE die Brücke zwischen WebClient und Desktop Client sein, ein Desktop Client mit WebService Funktionalitäten. Es gibt in einigen genannten Frameworks die Möglichkeit Webanwendungen in Desktopanwendungen zu portieren. Ob die Anwendung im Internet und auf dem Desktop läuft, hat jedoch dabei keinen Vorrang. Weiterhin wird versucht die Unterschiede und Gemeinsamkeiten der Rich Clients und Rich Internet Application, sowie Besonderheiten herauszuarbeiten.
Der Großteil der betrachteten Technologien steht für die studentische Arbeit frei zur Verfügung, um so die passenden Tools komplett verwenden zu können. Einige der in der Abschlussarbeit verwendeten Rich Client sind für den kommerziellen Nutzen kostenpflichtig, was im späteren Verlauf Bemerkung findet.

In der heutigen Zeit findet eine rasante Entwicklung in der IT Branche vor. Dementsprechend müssen auch die Anwendungen schneller und effizienter entwickelt werden. Dabei sollen beispielsweise Frameworks verschiedenster Art helfen, in dem vorgefertigte Basisfunktionalitäten mitgeliefert werden, das sind in der Regel graphische Komponenten und grundlegende Services. Somit muss eine Anwendung nicht von Grund auf neu programmiert werden. Allgemein sind Frameworks so angelegt die Anwendung leichter erstellen zu können. Damit soll der Fokus auf die eigentliche Arbeit gelenkt werden, der Programmierung der Logik einer Anwendung.

3.1. Definition

In diesem Abschnitt werden die grundlegenden Merkmale eines Rich Client und einer Rich Internet Application aufgezeigt. Es soll dabei keineswegs als eine perfekte Definition gelten, weil dies aufgrund von unterschiedlichen Ansichten nicht möglich ist. Es ist anzumerken, dass hier verschiedene Definitionen zusammengetragen werden. Nimmt man die Definition von Heiko Böck, Autor des Buches „Netbeans Platform 6 – Rich Client Entwicklung mit Java“ [3], so ist ein Rich Client ein Teil einer Client-Server Architektur. Er übernimmt die Verarbeitung und stellt die grafische Oberfläche dar. Weiterhin ist ein Rich Client durch Module oder Plug-Ins erweiterbar, besitzt ein Update Mechanismus und ist auf Basis eines Frameworks aufgebaut. Hinzu kommt eine Plattformunabhängigkeit dessen. Es ist möglich on- und offline zu arbeiten.
Betrachtet man nun die Definition in Wikipedia [17] genauer, muss man sich dort mit einer kurzen Antwort begnügen. Als Kernaussage folgendes angeben: „Der Rich Client ist ein neuer Ableger des Fat Client mit reichhaltigeren Problemlösungen. Meist handelt es sich um ein Framework, das durch Module und Plug-ins erweiterbar ist. So kann ein Rich Client nicht nur ein Problem lösen (wie bei Fat Clients üblich), sondern ist auch für artverwandte oder gar artfremde Probleme geeignet. […]“. Auch wenn Wikipedia als kritisch in Bezug der Korrektheit angesehen wird, soll es nur als Beispiel genutzt werden, welches die unterschiedlichen Auffassungen des Begriffs Rich Client widerspiegelt.
Eine weitere Definition der Internetseite Kioskea.net [16] sieht andere Merkmale in einem Rich Client. Die Definition lautet dementsprechend, „Der « Rich Client »

ist ein Mittelweg zwischen dem Thin Client und dem Fat Client. Das Ziel des Rich Clients ist es daher, ein grafisches Interface zu bieten, geschrieben in einer Sprache, die auf der XML,-Syntax basiert, um Funktionen zu erhalten, die denen eines Fat Clients gleichen[…]Bei Rich Clients ist es möglich, die wichtigsten Verarbeitungsaufgaben vom Server ausführen zu lassen[…]“ Diese Definition könnte man auch schon als Rich Internet Application interpretieren. Man sieht also, dass eine Definition eines Rich Client sich nicht einfach gestaltet, da der Begriff keine einheitliche Bedeutung hat. Es gibt mehrere Auslegungen, jedoch keine wie man sie in einem Lexikon finden würde. Eines haben alle drei Beschreibungen gemeinsam; der Rich Client befindet sich lokal auf einem PC. Es ist also eine Anwendung, die die Verarbeitung der Daten auf dem Client PC ausführt und eine grafische Oberfläche stellt.
In den Definitionen wird mehrfach Fat und Thin Client erwähnt. So sollten noch die Begriffe erklärt werden. Beide Begriffe werden im Umfeld der Client-Server Architektur gesprochen. Grob umschrieben handelt es sich bei einem Fat Client um eine lokal installierte Anwendung, bei der sämtliche Aufgaben, wie Ein- und Ausgabe und die Berechnungen auf dem Client PC ausgeführt werden. Der Thin Client ist das genaue Gegenteil. Es ist eine Anwendung, die nur die Ein- und Ausgabe regelt, Berechnungen erfolgen auf dem Server.

Zusammengefasst besitzt ein Rich Client folgende Merkmale:

- Ist ein Client in einem Client-Server Verhältnis
- Ist lokal auf einem PC installiert
- Nutzt ein Netzwerk um Daten auszutauschen
- Möglichkeit on- und offline zu arbeiten
- Verarbeitung der Daten erfolgt auf dem Client PC
- ermöglicht eine komplexe Oberfläche
- Besitzt meist ein Framework
- Besitzt ein Komponentenmodell und ist erweiterbar
- Plattformunabhängig

Die Unterschiede des Begriffes rühren aus den verschiedenen angebotenen Frameworks. Anwendungsentwickler haben oft eine favorisierte Plattform, von

der aus entwickelt wird. So entstehen auch unterschiedliche Ansichten darüber, was ein Rich Client leisten soll.

Die Definition einer Rich Internet Application gestaltet sich einfacher. Der Begriff ist das erste Mal im Zusammenhang mit der damals zu Macromedia gehörenden Flex Umgebung aufgetaucht. Auf mehreren Internetseiten [38-40] findet man ähnliche „Definitionen“. Übereinstimmend können folgende Merkmale genannt werden:

- im Browser ausführbar
- Plattform- und browserunabhängig
- Keine Installation nötig - meist nur ein Plug-In benötigt
- bieten ähnliche Bedienfunktionalitäten wie Desktop Anwendungen
- Interaktion mit dem Nutzer
- Verbindung mit dem Internet benötigt
- Laufen meist in einer Sandbox[4]
- Kann Teile der Verarbeitung auf den Client auslagern

Jedoch kann man folgende Punkte ebenfalls einer RIA zuschreiben.

- meist mit Hilfe eines Frameworks aufgebaut
- Einfache Verteilung der Anwendung an Anwender

[4] Anwendungen werden in einem isolierten Bereich ausgeführt, welcher beschränkten Zugriff auf das eigentliche Client System und deren Hardware gewährt

Ein gutes Beispiel für eine Rich Internet Application ist Picnik [20]. Es ist ein einfaches Fotobearbeitungsprogramm komplett in Adobe Flex geschrieben. Hier ein Ausschnitt.

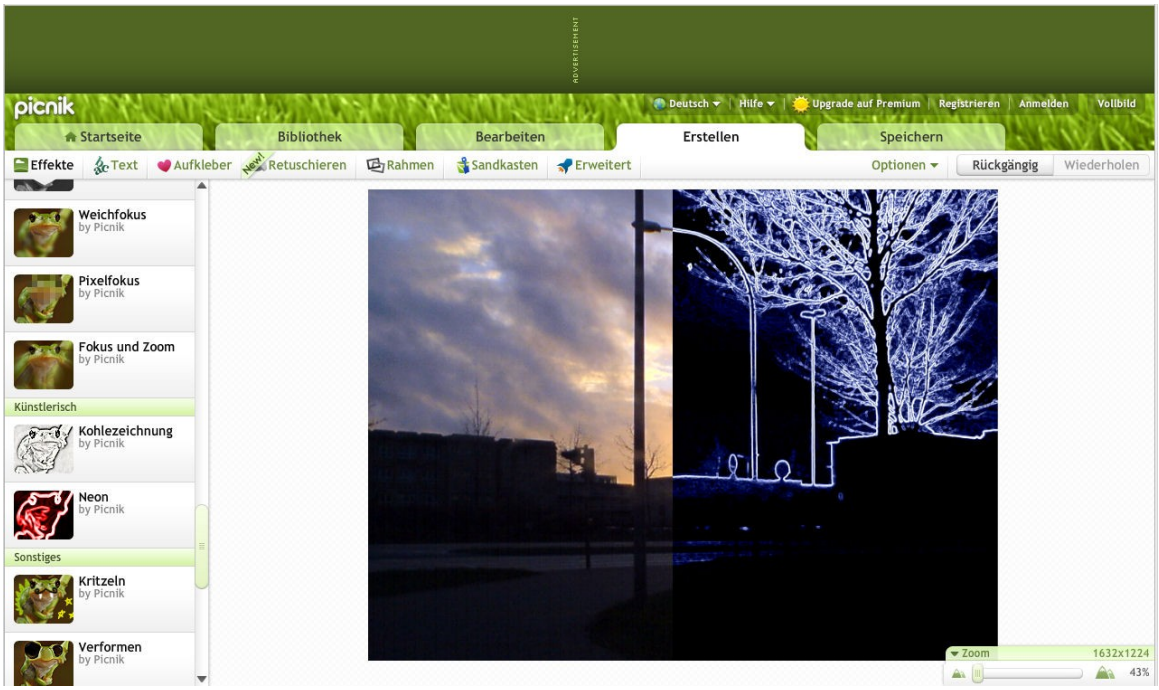

Abbildung 7 – Picnik (links vor, rechts nach der Bearbeitung eines Bildes)

Es ist eine einfach zu nutzende Webanwendung. Jedoch bemerkt man nach längerem Nutzen und mehreren bearbeiteten Bildern, dass die Anwendung überfordert wird. Die Berechnungen der Effekte dauern länger als bei einem ähnlichen Desktop Pendant. Es bleibt aber ein nützliches Tool.

3.2. Allgemeines

Dank Weiterentwicklungen in allen Bereichen der IT Branche, sei es Hardware oder Software, sind Rich Clients so wie Sie hier präsentiert werden möglich. Blickt man zurück zu den Anfängen von Java und vergleicht die Definition des Rich Clients, so erkennt man, dass Java viele dieser Punkte erfüllt. Rich Clients haben sich jedoch nicht durchgesetzt. Gründe waren vorrangig die schwache Hardware und die noch nicht so leistungsfähige Java Umgebung. Mit zunehmender Popularität des Internets wurden somit auch Thin Clients genutzt. Rich Internet Application waren zu dieser Zeit nicht denkbar, es fehlten die Technologien und Bandbreite in der Übertragung der Daten über das Internet.

Diese Mankos treffen heute nicht mehr zu, neue Techniken, Programmierwerkzeug und Weiterentwicklungen im Soft- und Hardwarebereich machen es möglich. Die steigende Anzahl an Rich Internet Application zeigt es.
Ein heutiger Rich Client respektive eine Rich Internet Application Framework zeichnet sich auch durch eine Reihe von Widgets[5] und Services aus. Beides sind vorgefertigte Komponenten, die eine Wiederverwendung für mehrere Projekte erlauben, alles im Sinne einer einfacheren Entwicklung von Anwendungen. Dazu trägt auch das einmalige Schreiben des Codes für alle unterstützten Plattformen bei. Wichtig ist die Verbreitung des Frameworks auf möglichst viele Plattformen.
Im Kern sind Rich Clients und Rich Internet Application zwar Fat- und Thin Clients, was aber einen entscheidenden Unterschied ausmacht, sind die Frameworks, welche eine einfachere und schnellere Entwicklung von Anwendungen ermöglichen soll. Ferner bieten einige Technologien die Möglichkeit Anwendung für Desktop und das Web zu erstellen. Beispielsweise unterstützen Silverlight und RAP diesen Mechanismus. Im Kontext der Client-Server Architektur steht immer die Verteilung der Rechenlast und des Kommunikationsaufkommen. Ziel einiger RIA ist die Rechenlast möglichst gleichmäßig auf Client und Server zu verteilen. So kann man neben grafischen Operationen auch einige Anwendungsberechnungen auf den Client auslagern.

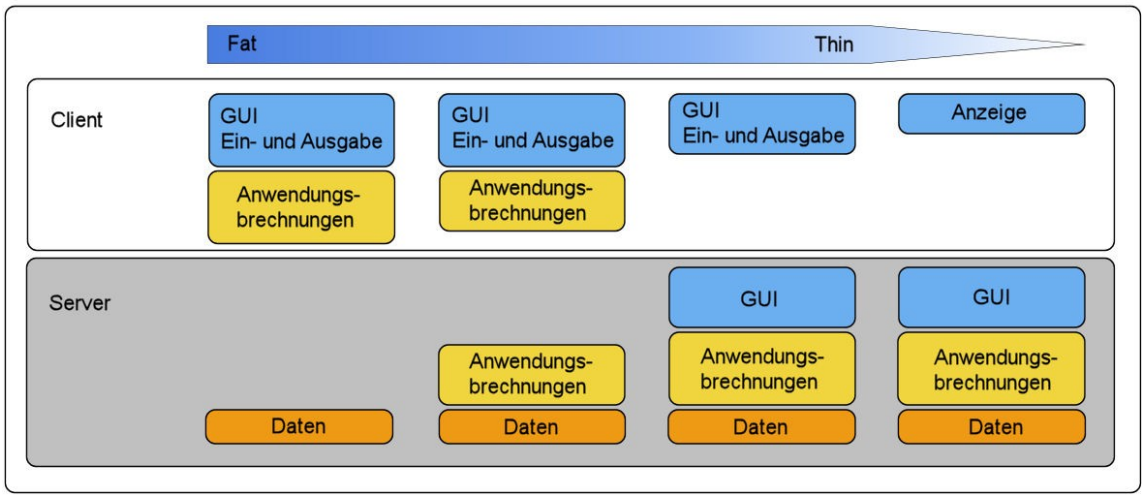

Abbildung 8 – Verteilung der Rechenlast in Client-Server Architektur [25]

Die Grafik verdeutlicht oben beschriebenen Aspekt. Welche Strategie gewählt wird ist von Framework zu Framework unterschiedlich. EclipseRCP kann am

[5] Sind im Bereich der Benutzeroberflächen grafische Steuer- und Bedienelemente oder kleine Hilfsprogramme, die die Anwendungsentwicklung erleichtern sollen

ehesten zu den Fat Clients zugeordnet werden. Die Rich Ajax Platform kann dagegen der dritten Spalte zugeordnet werden. Das Framework lässt grafische Elemente auf dem Server, sowie auf dem Client erzeugen. Mehr dazu kann in Kapitel 3.3.5 erfahren werden. Die restlichen hier behandelten Technologien ordnen sich zwischen „Fat“ und „Thin“ ein. Die letzte Spalte ist den Ultrathin Clients vorbehalten, bei denen der Client nur für die Anzeige von Daten genutzt wird. Eine gleichmäßige Verteilung kann auch das Datenaufkommen zwischen Client und Server reduzieren. Ein Rich Client bietet hier trotzdem Vorteile. Durch sämtliche Berechnungen auf dem Client PC sind Anwendungen in der Regel leistungsfähiger und reaktionsschneller, was auch mit einer reduzierten Kommunikation zwischen Client und Server verbunden ist. Dafür wird jedoch mehr Rechenleistung als bei einem Thin Client gefordert. Zudem ist eine lokale Installation bei Verwendung von Fat Clients erforderlich, wobei Rich Internet Application meist nur ein Plug-In zur Ausführbarkeit benötigen, oder wie Thin Clients keine weiteren Maßnahmen nötig sind. Es müssen also auch weiterhin grundlegende Entscheidungen in Bezug auf Architektur der entstehenden Anwendung getroffen werden.

3.3. Ausgewählte Rich Clients und Rich Internet Applications

Bei den Entscheidungen sollen die nun folgenden Seiten helfen. Aus einer Vielzahl von Frameworks am Markt, wurden fünf Kandidaten ausgewählt und werden vorgestellt.

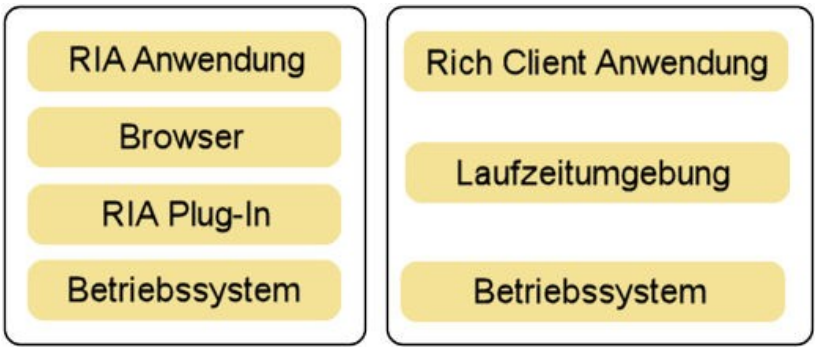

Abbildung 9 – Aufbau von RIA und Rich Client Architekturen

Die Grafik stellt eine vereinfachte Form der benötigten Komponenten dar. Zu Grunde liegt das Betriebssystem. Die nächste Schicht bilden die benötigten Komponenten um RIA Anwendungen ausführen zu können. Für die

ausgewählten Kandidaten Silverlight, JavaFX und Flex trifft dieses Modell zu. Die Rich Ajax Platform benötigt derweil keine Plug-Ins. Es setzt vollkommen auf im Browser integrierte Techniken, namentlich AJAX. Die Plug-Ins werden systemweit installiert und stehen für alle Browser zur Verfügung. Die Browser dienen dabei als Schnittstelle zwischen Mensch und Maschine, zur Ein- und Ausgabe und Anzeige der Anwendung. Die Anwendungen werden für gewöhnlich im Browser ausgeführt, aber einige unterstützen auch die Nutzung außerhalb dieser. Adobe Flex benötigt dafür eine eigene Laufzeitumgebung, Silverlight funktioniert wiederum ohne weitere Plug-Ins oder ähnlichem als Desktopanwendung.

Der Aufbau der Rich Client Anwendungen weißt einige Unterschiede auf. Im Gegensatz zu Rich Internet Application wird nicht nur ein Plug-In benötigt, sondern eine eigene Laufzeitumgebung. Diese ermöglicht die Ausführung auf dem Client PC. Die Rich Client Anwendungen sind lokal an den Client PC gebunden.

3.3.1. JavaFX

JavaFX ist im Vergleich zu Adobe Flex und Microsoft Silverlight das jüngste RIA Framework. Es hat jedoch für Aufmerksamkeit gesorgt, da Sun Microsystems es als direkte Konkurrenz zu oben genannten Technologien platziert hat.

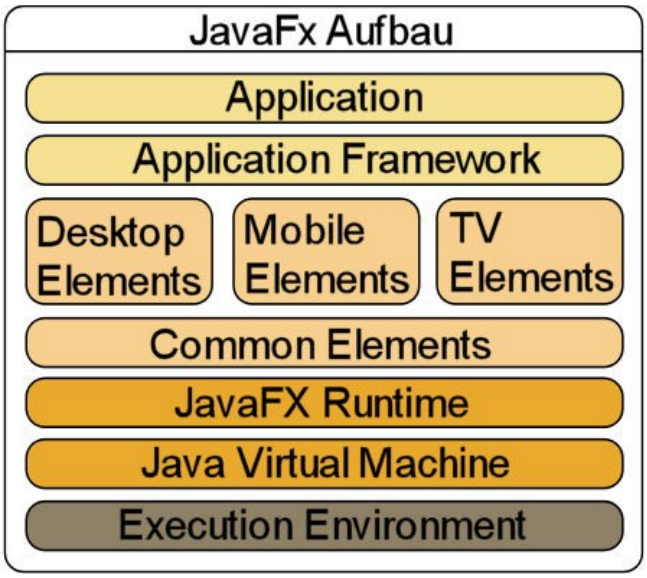

Abbildung 10 – Aufbau der Architektur in JavaFX [9]

Das JavaFX Framework ist nicht nur für den PC Bereich konzipiert. Es unterstützt ebenfalls den Mobilbereich und in kommenden Releases wird auch der TV Bereich integriert. JavaFX baut auf der Java Virtual Machine auf und zieht daraus mehrere Vorteile. Java selbst genießt eine weite Verbreitung[34]. Somit haben JavaFX Anwendungen eine solide Verbreitungsgrundlage. Das JavaFX Plug-In wird beim ersten Start einer solchen Anwendung automatisch herunterladen und installiert. Ein weiterer Vorteil ist die Bindung an die JavaVM. Es ist möglich normale Java Bibliotheken in JavaFX Code einzubinden. Die Anwendungen werden in JavaFX Script geschrieben. Die Skriptsprache verwendet einen deklarativen Programmieransatz. Unterhalb der JavaFX Runtime und der JavaVM liegt die Betriebssystemschicht. Unterstützt werden die Windows Betriebssysteme ab Windows XP, das MacOS X 10.4.1, seit Version 1.2 auch Linux und Solaris.

JavaFX versucht vor allem die Erstellung von Grafischen Oberflächen zu erleichtern. So können Grafiken in JavaFX Code konvertiert werden, konkret gibt es, als Teil der JavaFX Production Suite Plug-ins, für Adobe Photoshop und Adobe Illustrator. Schaut man in die API[6] [27] erkennt man die Ausrichtung auf Grafikmanipulation und –effekten, sowie Animation. Ein wichtiges Grundelement zur Animation in das *Timeline* Element, welches die Animationen gesteuert werden. In diesem Fall sind es die Scenes, die verändert werden. Sie können zeit- und eventgesteuert sein. Weiterhin enthält JavaFX viele vorgefertigte Effekte. Neben den Effekten findet man auch verschiedene Steuerelemente vor, wie Button und Slider.

Zurück zur oberen Grafik. Die Common Elements werden für alle Plattformen eingesetzt wie 2D Grafik, Animation, Texte, Audio- und Videosupport, Netzwerk oder auch Zugriffe auf lokalen Speicher. Sie bilden das Grundgerüst. Für die verschiedenen Anwendungsgebiete gibt es spezielle Komponenten. Hierzu zählen die Desktop-, die Mobile- und TV Elements. Erstere enthalten Funktionen wie den Application Lifecycle Support, einen Swing Wrapper, XML/JSON Support oder auch Zugriffe auf Datenbanken.

Da JavaFX noch ein relativ neues Projekt von Sun Microsystems ist, kann die Performance in einigen Anwendungsbereichen im Verhältnis zu Flash oder Silverlight geringer sein. Es wird ständig an der Leistungsfähigkeit gearbeitet.

[6] Application Programming Interface

Laut Sun Microsystems wurde die allgemeine Leistung von Version 1.0 zu 1.2 gesteigert [28]. Es fanden in jeder Version Änderungen an der API statt, eine Abwärtskompatibilität kann nicht gesichert werden. Für bestehende JavaFX-Anwendungen ist dieser Fakt problematisch. Sie können nicht weiter verwendet werden, müssen angepasst und neu kompiliert werden. Ob weitere Änderungen zu erwarten sind lässt sich nicht abschätzen.

3.3.2. Microsoft Silverlight

Microsoft Silverlight ist wie JavaFX auch ein Framework um Rich Internet Application zu erstellen. Es steht in direkter Konkurrenz zu Adobe Flex und JavaFX. Das Software Development Kit für Silverlight steht kostenfrei zur Verfügung. Jedoch sind die offiziellen Entwicklungswerkzeuge Visual Studio und Expression Blend nicht frei erhältlich, weiter Kosten fallen für die Microsoft Server an.

Zum Ausführen der Anwendungen ist nur das Browser Plug-In nötig, das heißt, dass keine weitere Softwareinstallation notwendig ist. Das Plug-In ist für Windows XP/Vista/7 und MacOS verfügbar. Von inoffizieller Seite wird eine Portierung für Linux vorgenommen, namentlich das Moonlight Projekt. Zu den unterstützten Browsern gehören IE 6-8, Safari und Firefox. Es werden die gängigsten Betriebssysteme unterstützt. Man kann von einer teilweisen Plattformunabhängigkeit sprechen, da die Linux Portierung nicht von offizieller Seite unterstützt wird, man nicht auf eine volle Kompatibilität setzen kann und man allein von der Unterstützung des Moonlight Projektes abhängig ist.

Anwendungen werden in XAML und .NET programmiert. XAML steht für Extensible Application Markup Language. Damit werden sämtliche Elemente in Silverlight wie zum Beispiel geometrische Figuren, Buttons oder Bilder definiert. Dank der Unterstützung der .NET Common Language Runtime, kurz CLR, können Silverlight Anwendungen in jeder .NET Programmierumgebung geschrieben werden, die in der Lage ist die Silverlight Common Language Runtime anstatt der .NET CLR auszuführen. Mit Hilfe der in Silverlight verwendeten Implementierung der Dynamic Language Runtime können weitere Skriptsprachen, wie IronRuby und IronPython, angesprochen werden [44].

Microsofts Silverlight ist 2006 erschienen. Seit der Veröffentlichung ist Silverlight in Version 3 erschienen, Version 4 ist für nächstes Jahr geplant. Somit gibt es innerhalb von 3 Jahren drei Versionen. Die einzelnen Versionen sind in bestimmten Bereichen nicht kompatibel zueinander. Das merkt man, wenn man versucht Silverlight 2 Anwendungen mit installiertem Silverlight 3 Plug-In zu nutzen. Die Anwendung fordert den Nutzer auf Silverlight 2 herunter zu laden. Ist aber einmal eine höhere Version ins System eingespielt ist ein Downgrade beziehungsweise eine parallele Installation verschiedener Silverlight Versionen nicht möglich. Silverlight wurde dafür aber um viele Features erweitert. So ist es seit Version 2 möglich einzelne Teile des .NET 3.x Frameworks für Silverlight Anwendungen zu nutzen, Beschränkungen gibt es hingegen im Zugriff auf das Dateisystem des jeweiligen Betriebssystems. Mit Silverlight 3 wurden Funktionen implementiert, Anwendungen ohne größere Probleme als Browser- oder Desktopanwendung verwenden zu können [35]. Zudem wurden weitere vorgefertigte Elemente für die Gestaltung der Oberfläche hinzugefügt.
Silverlight besitzt jedoch noch keine entsprechende Verbreitung wie es bei Adobe Flash der Fall ist. Adobe Flash besitzt die höchste Verbreitung. Das liegt aber auch daran, dass Flash schon länger als Silverlight am Markt ist und dessen Erscheinen ein Quasi-Monopol im Bereich der Rich Internet Application inne hat. Wie bei JavaFX lässt sich nicht feststellen, ob mit fortschreitenden Versionen eine Abwärtskompatibilität erhalten bleibt, was die Anwendungsentwicklung recht unsicher macht.

3.3.3. Adobe Flex und Adobe AIR

Adobe Flex ist die Plattform mit der meisten Erfahrung im Bereich der Rich Internet Application. Flex ist eine Weiterentwicklung von Flash. Adobe Flex basiert auf der Flash Technologie und richtet sich dank ActionScript und MXML an Programmierer im eigentlichen Sinne. Flash hingegen ist grafik- und animationsorientiert und ermöglicht die Wiedergabe interaktiver Inhalte. Flex ist ursprünglich ein Produkt von Macromedia. Mit der ersten Version diente es als reiner Präsentationsserver und war ein kostenpflichtiges Produkt. 2004 wurde Macromedia von Adobe übernommen. Mitte 2006 erschien Flex 2 unter der Flagge des neuen Besitzers. Mit der Übernahme änderte sich die Richtlinie für

Flex. Das SDK wurde kostenlos zur Verfügung gestellt. Es unterliegt der MPL[7]. Version 3.0 der wurde im Februar 2008 veröffentlich. Aktuell ist die Version 3.3 erhältlich, Flex 4.0 ist bereits in Planung. Adobe Flex verwendet zwei Skriptsprachen in der Flex Umgebung. Zum einen ist es ein angepasster XML Standard, MXML genannt. Zum anderen wird ActionScript genutzt. Es ist eine von Adobe entwickelte Skriptsprache. Sie basiert auf dem ECMAScript Standard, auf welchem auch JavaScript aufsetzt. Momentan ist ActionScript 3 aktuell. MXML ist eine deklarative Sprache und dient der grafischen Gestaltung der Oberfläche, während mit ActionScript die eigentliche Anwendungslogik programmiert wird. Der mitgelieferte Compiler wandelt zuerst den gesamten MXML Programmcode in ActionScript um, wonach diese in das Flashformat(.swf) kompiliert wird.

Adobe Flex ist nur teilweise kostenlos. Dazu zählen das Software Development Kit und der Flash Player. Der Flex Builder muss gekauft werden. Dieser ist die Standard Entwicklungsumgebung für das Flex SDK. Er ist als Stand-Alone Variante und als Plug-In für Eclipse erhältlich. Die heutige Verbreitung des Flash Players wird auf 97% [34] alle internetfähigen PCs geschätzt, denn nur dieser und ein passender Webbrowser werden für die Verwendung von Flex-Anwendungen benötigt. Soll dieses Programm jedoch auf dem Desktop genutzt werden, muss es an die Adobe AIR Umgebung angepasst werden. Es bildet die Desktop Umgebung für Flex Anwendungen. Die Anwendungen laufen mit einigen Anpassungen außerhalb des Browsers. Adobe AIR bietet eine eigene API um Anwendungen zu erstellen. Diese bietet erweitere, auf den Desktop angepasste, Funktionen. Ein Beispiel ist der Zugriff auf das Dateisystem, Dateien können geschrieben werden. Oder aber man kann die in Adobe AIR integrierte SQLite Datenbank nutzen. Adobe Flash CS4, Adobe Dreamweaver und Adobe Flex Builder integrieren Adobe AIR und ermöglichen die Erstellung einer Desktopanwendung aus genannten Programmen heraus. Zur Wiedergabe von Inhalten und Anwendungen im Webbrowser wird der Flash Player benötigt.

Adobe bietet mit Flex eine plattformunabhängige Lösung. Der Flash Player und die Adobe AIR sind für Windows, Linux und MAC OS verfügbar. Unterstützte Browser des Flash Players sind Firefox, der Internet Explorer, Safari und Opera.

[7] Mozilla Public License

Will eine Rich Internet Application nutzen, die auf nahezu jedem PC läuft, ist man mit Adobe Flex auf der sicheren Seite.

3.3.4. Eclipse RCP

Die EclipseRCP ist eine Zweigentwicklung der Eclipse IDE. Sie basiert auf dem OSGi Spezifikationen und hat diese mit Eclipse Equinox implementiert. Im Allgemeinen besteht selbst EclipseRCP aus Plug-Ins. Sie ist komplett in Java geschrieben. Und bietet standardmäßig Java als Programmiersprache an. So ist es auch möglich die Standard Widget Toolkit Grafikbibliothek zu nutzen. Der Einfachheit halber wird die Bibliothek SWT genannt. Der große Vorteil gegenüber den anderen Plattformen ist die Erweiterbarkeit. Außerdem basiert es als Open Source veröffentlicht. Die gesamte Entwicklungsumgebung ist kostenfrei.
Mit der EclipseRCP steht die SWT Grafikbibliothek zur Gestaltung der grafischen Oberfläche zur Verfügung. Nun hat man mehrere Möglichkeiten die Anwendung zu gestalten. Zu Wahl steht das Standard Look and Feel der Eclipse Anwendung. Alternativ kann auch die reine SWT Bibliothek genutzt werden. Die Besonderheit besteht in der Unterstützung des Aussehens des jeweiligen Betriebssystems. Die SWT Bibliothek greift dazu auf die Grafikressourcen des Betriebssystem zu und nutzt diese um die Java Anwendung zu gestalten. Im Normalfall hat es Geschwindigkeitsvorteile gegenüber einer Swing Oberfläche, was auch der eigentliche Grund für die Entwicklung der SWT Bibliothek ist. Sind Elemente in den jeweiligen Betriebssystemressourcen nicht vorhanden, werden diese emuliert. Die dritte Möglichkeit ist das komplette Neuschreiben eines grafischen Stils. Wie die Oberfläche der Eclipse Programmierumgebung aussieht, verdeutlicht ein Bild. Es soll das grundlegende Design darstellen. Auffälligste Unterschiede gibt es in der Gestaltung der Fensteroptionen und der Tabs.

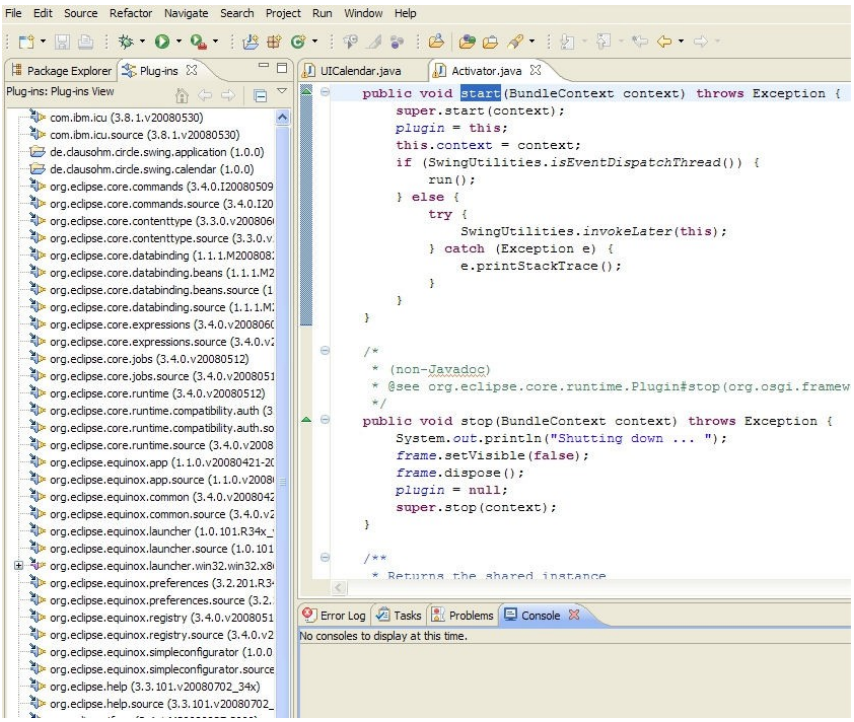

Abbildung 11 Ausschnitt der Eclipse IDE

Nun werden zum Vergleich die möglichen Aussehen einer Anwendung mit der SWT Bibliothek gezeigt.

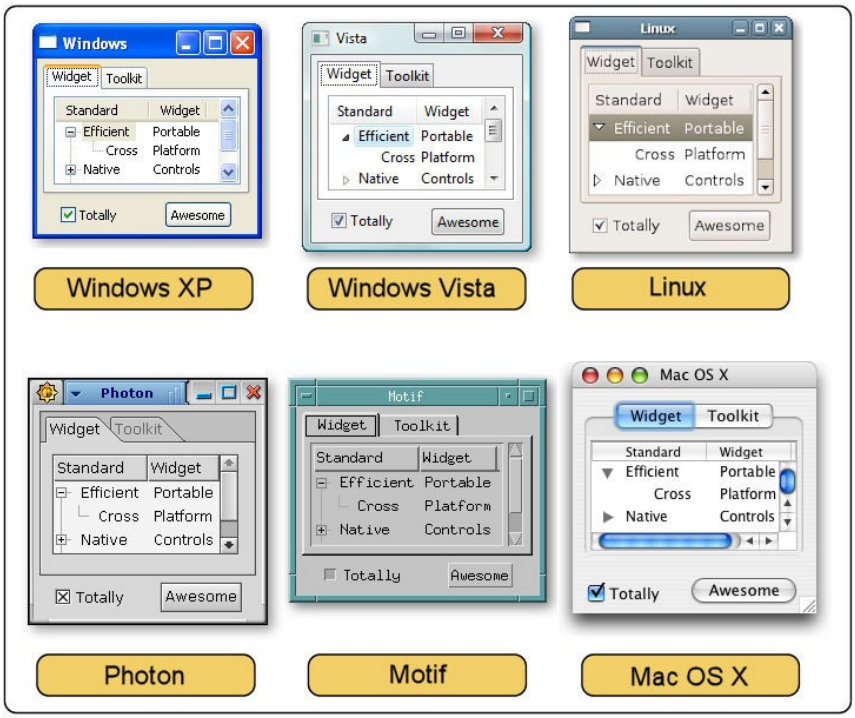

Abbildung 12 – SWT Look and Feels [31]

Die Entwicklung eines eigenen Aussehens nimmt die meiste Zeit der drei genannten Möglichkeiten in Anspruch. Dafür hat man eine den eigenen Wünschen angepasste Oberfläche.
Im Gegensatz zu allen anderen vorgestellten Kandidaten, ist die EclipseRCP eine reine Umgebung um Rich Client zu entwickeln. Also Anwendungen für den Client PC.

3.3.5. RAP – Rich Ajax Platform

RAP ist mittlerweile ein Projekt der Eclipse Foundation, also eine Open Source Lösung. Die Anfänge der Technologie wurden in der Firma Innoopract GmbH geschaffen. Es hat das Ziel Anwendungen als Desktop- und Webclient mittels der gleichen Programmiersprache bereitzustellen. Als einzige der ausgewählten Technologien benötigt es keine Plug-Ins oder sonstige Installationen. Das ist auf die Verwendung der durch 1&1 erstellten JavaScript Bibliothek qooxdoo zurückzuführen. RAP verwendet allein XML, JavaScript und HTML zur Generierung von Anwendungen. Die Rich Ajax Platform ermöglicht es bestehende Eclipse Anwendungen für das Internet zu portieren. Laut eines Artikels aus dem Java-Magazin [11] sollen Teile bestehender Anwendungen erneut verwendet werden können. Erstellt man neue Projekte mit RAP besitzt man die Grundlage Anwendungen sowohl für den Desktop als auch für das Web zu schreiben. Diese Art der Anwendungsprogrammierung wird auch SingleSourcing genannt. RAP ist unter der EPL Lizenz veröffentlicht. Diese lässt eine kommerzielle Verwendung zu.
RAP baut auf einem Java-basiertem Entwicklermodell auf, in diesem Fall nutzt es die Eclipse Plattform. Daraus ergibt sich einer der Vorteile der Rich Ajax Platform, die gleiche Codebasis. Weiterhin wird Eclipse Equinox genutzt, welches eine Implementierung des OSGi Standards darstellt. Auf OSGI wird im weiteren Verlauf der Arbeit eingegangen. Um jedoch vorzugreifen, die Module im OSGi Kontext werden Bundles genannt und ermöglichen eine Modularisierung oberhalb der Packages. Durch die Implementierung der OSGi Spezifikation wird die Nutzung von Eclipse Rich Client Bundles und Eclipse Plug-Ins ermöglicht und bereits erstellte Bundles können teilweise für eine RAP Webanwendung weitergenutzt werden. Mittels einer OSGi Servlet Bridge können die

Webanwendungen in einem Java Enterprise Edition Container, also Webservern wie Tomcat oder Jetty laufen.
Anders als bei der einigen Mitbewerbern, werden für RAP Anwendungen keinerlei Browser Plug-Ins oder Addon benötigt.

Eine RAP Anwendung hat einen ähnlichen Aufbau wie eine mit Eclipse RCP erstellte Anwendung.

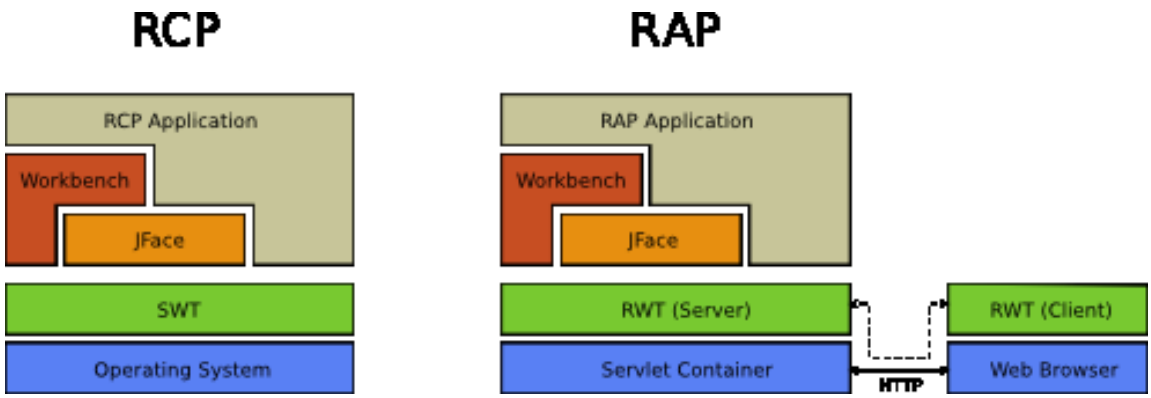

Abbildung 13 – Vergleich des Architekturaufbaus [18]

Sie besteht im Folgenden aus der RCP Applikation, der Workbench, JFace[8] und SWT der Betriebssystemschicht. Auffällig ist, dass die JavaVM als Laufzeitumgebung benötigt wird. Weitere Änderungen erfolgen, wenn man die Anwendung zur Webanwendung umbaut. So wird die SWT Bibliothek, durch eine Nachbildung dieser, ersetzt. Sie wird in Anlehnung an die SWT Bibliothek RAP Widget Toolkit bezeichnet, auch RWT genannt. Die Notwendigkeit dafür bestand in der fehlenden Webfähigkeit der SWT Bibliothek. Sie nutzt zur Darstellung der grafischen Oberfläche, wenn möglich, Betriebssystemeigene Komponenten um ein betriebssystemspezifisches Aussehen zu gewährleisten. Das RWT kann jedoch nicht alle Funktionen der SWT nachbilden. Neben der SWT liefert die RAP Plattform angepasste Versionen der JFace und Workbench Bundles. Diese stellen ebenfalls einen Großteil der originalen Bibliotheken dar, bieten aber keine vollständige Funktionalität. Durch die Anpassungen der Bundles sind auch Veränderungen an einigen Schnittstellen notwendig geworden. Werden bereits bestehende Bundles genutzt, müssen diese entsprechend angepasst werden. Plant man ein komplett neues Projekt mit RAP, welches für den Desktop und als Webanwendung entwickelt wird, sollte man sich auf die gegebenen Funktionen

[8] Grafikbibliothek für SWT, die eine erweitere Darstellung ermöglicht

der angepassten SWT, JFace und Workbench Bibliotheken stützen um eine einheitliche Programmiergrundlage zu verwenden. Die Lösungen müssen so gewählt werden, dass Sie mit allen Bundles arbeiten. Kann das nicht gewährleistet werden, müssen nicht einheitlich nutzbare Codeabschnitte getrennt und beispielsweise in OSGi-Bundles-Fragmente oder eigenständige Bundles ausgelagert werden. Die Fragmente sind wie normale Module aufgebaut, jedoch mit dem zu erweiterndem Bundle verknüpft. Zur Laufzeit wird das Fragment in das Bundle integriert. Als Beispiel lässt sich hier ein Bundle, welches in einer Webanwendung Verwendung findet, mit Hilfe von Bundle Fragmenten, für den Desktopgebrauch erweitern. So wird dieses Fragment nur für die Desktopanwendung ausgeliefert. Für den Aufbau eines Bundles oder Fragmentes sei auf das Kapitel 4.3 verwiesen.

Bei einer parallelen Nutzung einer Anwendung für den Desktop und das Web treten weitere Besonderheiten auf. Ist eine gewöhnliche Eclipse RCP Anwendung nur auf einen aktiven Nutzer ausgerichtet, haben Webclients oft mehrere zeitgleiche Zugriffe auf das Programm durch verschiedene Benutzer. So muss auf eine korrekte Instanzierung der Funktionen geachtet werden. Es werden entweder Instanzierungen einer Funktion für jeden Benutzer oder eine einzelne anwendungsweite Funktion, die für alle Anwender gilt, benötigt. Bei Desktopprogrammen spielt dieses Verhalten keine Rolle, da dort nur ein aktiver Benutzer angemeldet ist. Für den Fall, dass Funktionen, die pro Nutzer benötigt werden, bietet das RAP Projekt session-bezogene Funktionalitäten. Diese Funktionen werden im Rahmen der Rich AJAX Platform Session-Singleton genannt. Zum Beispiel sind Nutzereinstellungen zu nennen, ohne eine Session Instanz würden alle Nutzer auf die gleichen Einstellungen zurückgreifen. Im Bereich der Grafikressourcen werden keine mehrfachen Instanzen verwendet. Erstellt man für jeden Benutzer eigene Ressourcen, würde der Speicherverbrauch zunehmen. Deshalb wird nur ein globaler Aufruf für alle User gewährt. Die Grafikklassen besitzen keine Konstruktoren und keine *dispose* Funktion. Der SWT Ersatz RWT nutzt dafür eine Hilfsklasse Graphics. Instanzen werden über Factory-Methoden erzeugt. Hier müssen für zu portierende RCP Anwendungen Codeänderungen vorgenommen werden. Für neue Projekte, die sowohl für den Desktop, als auch als Webanwendung entwickelt werden, kann

man sich mit in JFace integrierten APIs behelfen. Diese sind für RCP und für RAP Anwendungen vorhanden.
Eine RAP Webanwendung ist von der Architektur her ein verteiltes System aus Server und Thin Client. Das bedeutet, dass jedes Widget ein Server und einen Client-seitigen Teil besitzt. Die Verarbeitung findet somit auf dem Server statt. Sie kommunizieren über einen AJAX Request. Im Falle eines „*Button pressed*“ Event wird die Aktion an den Server geschickt, dort verarbeitet und die Antwort zurück geschickt, der Button wird als „*pressed*“ dargestellt und ein bestimmtes Event wurde ausgelöst.
SWT und RWT besitzen einen wesentlichen Unterschied in der Verarbeitung von häufig auftretenden Events, wie die Benutzereingaben(modify- und verify-Event). Während SWT jedes Event sofort abarbeitet, werden in RAP Anwendungen die Events durch einen *EventListener* gesammelt und in regelmäßigen Abständen an den Server gesendet und abgearbeitet. Daraus ergeben sich minimale Verzögerungen in der Ausgabe. Bei falschen Eingaben durch den Nutzer wird dieser also verzögert benachrichtigt. Weiterhin wird in RWT auf *MouseOver* Events zu Gunsten des Trafficaufkommens verzichtet. Die Funktionen *SWT PaintListener* und der *Graphic Context* sind noch nicht in der RWT Bibliothek umgesetzt. Diese werden für das Zeichnen direkt auf der Oberfläche von Widgets oder für grafische Editoren genutzt. Dies ist ein Problem wenn man zeichenintensive Anwendungen erstellen oder portieren will.
Die Funktion *FileDialog*, die zum Öffnen und Speichern von Dateien verwendet wird, steht für RAP Webanwendungen nicht zur Verfügung. Es ist nicht regelt ob auf das Client- oder auf das Serverdateisystem zugegriffen werden soll. Als Ausweg müssen die Up- und Download Funktionen des Browsers genutzt werden.
Für die Portierung komplexer Eclipse RCP ist RAP differenziert zu sehen, da es viele Anpassungen und Kompromisse gibt. Dank der OSGi Implementierung können viele Bundles von vorhandenen RCP Projekten weiterverwendet werden. Eine Portierung beschränkt sich aber wiederum nur auf mit Eclipse erstellte Projekte. Startet man ein neues Projekt und berücksichtigt die gegebenen Beschränkungen beziehungsweise die Besonderheiten des RAP Frameworks, dem steht ein an das Eclipse Entwickler Modell angelehntes „Tool“ zur Verfügung.

3.4. Vergleich der Technologien anhand von Kriterien

Der Vergleich der Rich Clients und Rich Internet Applications dient der besseren Übersicht der Merkmale und der Auswahl einer endgültigen „Technologie“ für den CIRCLE RC. Die folgend aufgelisteten Stichpunkte sind als allgemeine Leitpunkte zu verstehen. Die „Technologie“ soll idealerweise alle Punkte erfüllen. Wünschenswert ist eine:

- Weitestgehende Betriebssystemunabhängigkeit
- Nutzung im Webbrowser und als Desktopanwendung
- Zukunftssichere Technologie
- Erneuerung von Komponenten zur Laufzeit
- Verwendung eines eigenen Designs/eigener Grafikelemente
- Geringerer Wartungsaufwand
- Erweiterung der Funktionen und eine Aufbereitung der Auswertungen/Statistiken
- Mehrwert bei der Erstellung von Projekten

Um die Auswahl eines Frameworks zu erleichtern, erfolgt in der Vergleichsmatrix eine weitere Auswahl an Merkmalen.

	Silverlight	Flex
Anbieter	Microsoft	Adobe
Technologien	Silverlight und .Net[1]	ActionScript/MXML
RC/RIA*	RIA	RIA
Desktop/Webbrowser	beides	beides[2]
OSGI	nein	nein
Standard	Proprietär	Proprietär
Plattform	Windows und Macintosh inoffizielles Linux Plug-in	Windows, Linux und Macintosh
Kosten	offizielle Entwicklertools	offizielle Entwicklertools
Support	Microsoft	Adobe
Verbreitung	Gering (27%*)	sehr hoch (98%*)
erster Release (1.0)	April 2007	März 2004
letztes Update	März 2009	August 2009
Version	3.0	3.4

Tabelle 2 – Vergleich Teil 1

	Rich Ajax Platform	JavaFX	EclipseRCP
Anbieter	Innoopract GmbH Eclipse Foundation	Sun Microsystems	Eclipse Foundation
Technologien	AJAX/JavaScript	JavaFX Script	Java + SWT Bibliothek
RC/RIA*	RIA	RIA	RC
Desktop/Browser	Webbrowser	Webbrowser	Desktop
OSGI	ja	nein	ja
Standard	Open Source	Teilweise Open Source	Open Source
Plattform	Betriebssystemunabhängig Browserunabhängig[3]	Windows, Linux	Windows, Linux, Macintosh, Solaris
Kosten	Support durch Innoopract	-	keine
Support	Innoopract GmbH Eclipse Foundation	Sun Microsystems	Eclipse Foundation
Verbreitung	-	-	hoch (Java) (75%*)
erstes Release (1.0)	Oktober 2007	Dezember 2008	Juni 2004 (3.0)[4]
letztes Update	August 2009	September 2009	September 2009
Version	1.2.1	1.2.1	3.5 SR1

Tabelle 3 – Vergleich Teil 2

Einige Einträge sind mit Hinweisnotationen versehen. Nachfolgend sind die Erklärungen gegeben. Zu [1] lässt sich vermerken, dass Silverlight nicht den vollen Umfang des .NET Frameworks nutzt. Die [2] bezieht sich auf Adobe Flex. Mit dem Framework erstellte Anwendungen lassen sich als Desktop und als Webanwendung nutzen. Dazu muss jedoch das Adobe AIR Framework auf dem Client P installiert sein. Die dritte ([3]) Ergänzung muss beim Merkmal der unterstützten Plattformen gemacht werden. In diesem Fall betrifft es das RAP Framework. Auf Grund von nicht einheitlich umgesetzten Standards in den einzelnen Browsern kann zu unterschiedlichen Darstellungen der Anwendung führen. Das erste Release der EclipseRCP trägt die Versionsnummer 3.0. Da die OSGi Spezifikation erst in genannter Version implementiert wurde und somit der Grundstein für die Rich Client Platform gelegt wurde. Die in der Tabelle mit „*“ gekennzeichneten Prozentangaben beziehen sich auf Ergebnissen der Internetseite www.riastats.com.

Zu einigen verglichenen Merkmalen sind keine Daten vorhanden beziehungsweise sind keine fundierten Aussagen möglich. Diese Felder sind mit einem „-“ gekennzeichnet.
Im Laufe der Arbeit hat der OSGi Standard eine größere Bedeutung in der Firma erhalten. Man will ihn für kommende Anwendungen nutzen. Er hat für die spätere Auswahl eine größere Gewichtung erhalten, wodurch jedoch drei Technologien aus dem Raster fallen.
Viele der hier betrachteten Technologien sind nicht als Standard festgelegt. Hier ist vor allem auf die aktive Community bzw. Anbieter des Frameworks und die Verbreitung zu achten. Was nützt das beste Framework, wenn sich niemand dafür interessiert. Selbst wenn Standards definiert sind, wie zum Beispiel für JavaScript, legt dies nicht zugrunde, dass man auch noch Jahre später diese Technologie nutzt. Grund hierfür ist die schnelle Entwicklung der Technologien. Das bedeutet gleichermaßen, dass man nicht immer die neuste Technologie, die der Markt hergibt, besitzt und einsetzen kann. Wichtiger ist eine weite Verbreitung der Technologie. Betrachtet man Java, so kann man von einer weiten Verbreitung der Programmiersprache reden, egal ob im Mobilbereich, bei Web- oder Desktopanwendungen. Java wird vom Hersteller SUN Microsystems und durch den Java Community Process gepflegt. Teile der Programmiersprache sind als Open Source Projekt veröffentlicht. Somit ist gewährleistet, dass regelmäßig Updates und Implementierung neuer Features erscheinen, so wie es mit dem Java 6 Update 10 geschehen ist [21]. Anfang nächsten Jahres erscheint auch Java 7, welches ebenfalls Neuerungen bringen wird, beispielsweise wird die Swing Bibliothek erneuert, das Swing Application Framework und ein neuer Garbage Collector eingeführt [22].
Allgemein lässt sich zur Sicherheit der Technologien folgendes aussagen. Wird auf eine neue Technologie umgestellt, muss nicht nur eine Prüfung des Sicherheitskonzepts der Anwendung selbst, sondern auch die verwendete Technologie geprüft werden. Man begibt sich hier in die Hände eines Anbieters und deren Technologie. Die Anwendungen laufen zwar in einer Sandbox und bieten so einen gewissen Schutz, da es nur einen beschränkten Zugriff auf das System gibt, aber Sicherheitsrisiken im Browser oder mögliches Einschleusen von Schadcode durch Lücken in der Technologie sind möglich. So muss man immer darauf hoffen, dass die Anbieter Lücken schnell beseitigen.

3.5. Auswahl einer finalen Technologie

Nach Auswertung der gegebenen Daten, ist die EclipseRCP die beste Wahl. Die Plattform erfüllt die meisten Anforderungen, die für den CIRCLE RC gewünscht werden. Das wichtigste Kriterium ist die Unterstützung des OSGi Standard in Form der Equinox Implementierung. Dadurch werden die Updatefunktionalitäten und das Komponentenmodell gewährleistet, sei es für Eclipse selbst oder für damit erstellte Anwendungen. Fördernd wirkt sich auch die Tatsache aus, dass Eclipse eine komplette Programmierumgebung ist, die eine hohe Verbreitung hat. Allein die Eclipse 3.4 SR2 Versionen wurden, laut Angaben der Eclipse Homepage [26], in der gesamten Zeit mehrere Millionen Mal heruntergeladen. Es bietet ein Komplettpaket, eine IDE[9] und das Framework um Rich Client Anwendungen zu entwickeln. Die meisten anderen getesteten Frameworks besitzen nur ein SDK[10] und die Entwicklertools müssen einzeln ausgewählt werden. Eclipse ist Open Source, also kostenlos verfügbar. Standardmäßig wird Java genutzt, somit ebnet es den Weg für die Nutzung des Message Brokers ActiveMQ, welcher auf Java aufbaut. Durch die Verwendung von Java wird eine weitestgehende Plattformunabhängigkeit erlangt.

[9] IDE – Integrated Development Environment
[10] SDK – Software Development Kit

4. Erweiterung von CIRCLE

Das Kapitel begutachtet die Veränderungen, die im Rahmen der Umgestaltung von CIRCLE stattfinden. Es befasst sich mit der grafischen Oberfläche, dem OSGi Standard und dem Message Broker ActiveMQ. Letztere werden in ihrer Funktionsweise genauer gesichtet. Für die grafische Oberfläche werden zwei Layout Layouts erstellt.

4.1. Voraussetzung

Der CIRCLE Rich Client wird neue Anforderungen an das PC System setzen. Der Client wird mit Hilfe von EclipseRCP und der Java Technologie erstellt. Es wird also ein PC benötigt, auf welchem das Java Runtime Environment installiert ist. Kern ist die Java Virtual Machine. Die Wahl des Betriebssystems wird nur durch eine entsprechende Portierung des JRE[11] beschränkt. Sollen der Rich Client und das ursprüngliche CIRCLE System parallel auf einem Rechner genutzt werden, müssen die Systemanforderungen des CIRCLE Systems an den PC erfüllt sein. Hierbei kann nur Windows NT, 2000 und XP genutzt werden.

Wie bereits erwähnt, wird die Kommunikation zwischen den Anwendungen erneuert. Diese Veränderung muss beim Rich Client berücksichtigt werden. Es wird ein Nachrichtenverteilerdienst eingeführt. In diesem Fall Apache ActiveMQ. Dadurch wird ein einheitlicher Nachrichtenaustausch ermöglicht. Damit ein Datenaustausch stattfinden kann, muss das ganze System angepasst werden. Der Rich Client muss den JMS Standard beherrschen, aber auch das CIRCLE System muss mit dem Message Service umgehen können. Dazu wird ein weiteres Projekt der Apache Foundation, ActiveMQ CPP genannt, genutzt. Es gewährleistet die Anbindung von C/C++ Applikationen an ActiveMQ mittels CMS. Der C Message Service ist ein an den Java Message Service angelehnter Standard.

Damit ActiveMQ genutzt werden kann, muss ein Java-fähiger Server und die Java Runtime auf dem entsprechenden Betriebssystem installiert werden. Sind alle Voraussetzungen erfüllt, ergibt sich folgendes Schema.

[11] Java Runtime Environment

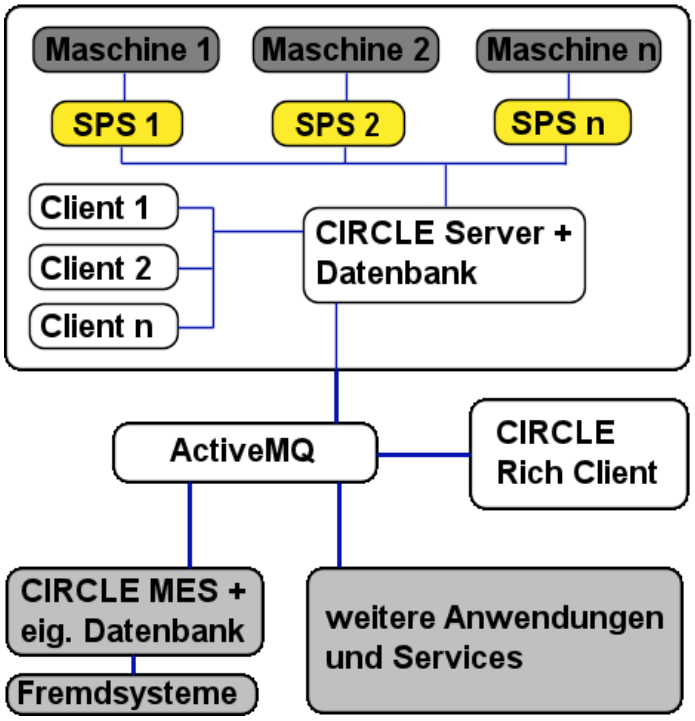

Abbildung 14 – Kommunikation mit ActiveMQ

Wie zu erkennen ist, bleibt das originale CIRCLE System im Grunde unangetastet. Es funktioniert weiterhin. Es müssen jedoch Anpassungen an der Serveranwendung vorgenommen werden. Wie im vorherigen Abschnitt erwähnt, muss eine Anbindung der C Applikation an den in Java geschriebenen Message Broker ActiveMQ stattfinden. Natürlich erfolgt eine Anpassung am Rich Client, sowie am CIRCLE MES. Es werden entsprechende Services programmiert um eine Verbindung mit ActiveMQ aufnehmen zu können. In Kapitel 4.3 werden weitere Details beschrieben.

4.2. Layout GUI

Da die Wahl des Kandidaten auf die EclipseRCP und somit Java und als Grafikbibliothek SWT gefallen ist, werden nun Konzepte für ein Layout des „CIRCLE Rich Client“ erarbeitet. Es werden zwei Layouts vorgestellt, hier nur als Grundrisse dargestellt. Das Hauptaugenmerk soll auf in der Anordnung der Elemente liegen. Beispiel 1 behält den Aufbau des originalen CIRCLE Analyzers weitestgehend bei. Die Titelleiste entspricht jetzt dem Windows Standard(1). Die ehemals in der Kopfleiste verankerten Funktionen Drucken, Export, Aufgabenliste, Kalender und

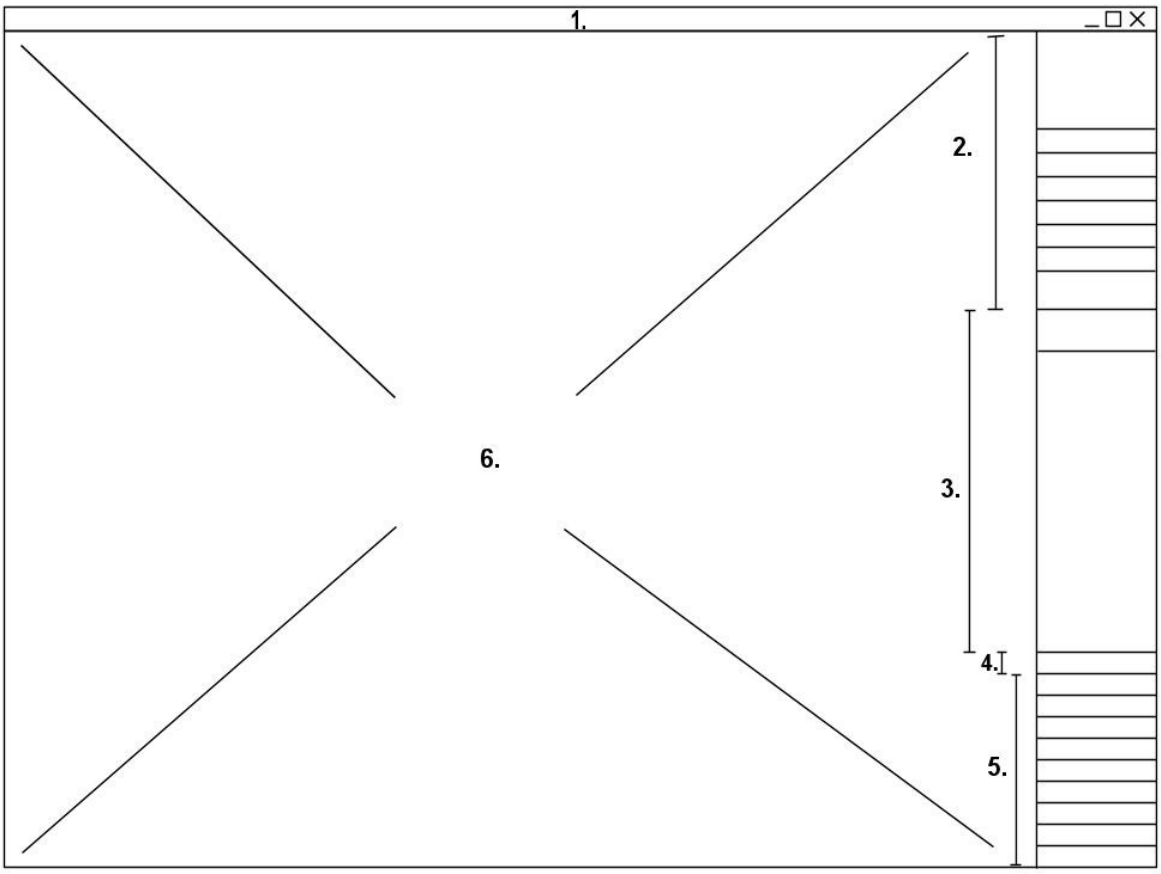

Abbildung 15 – Entwurf eines Layouts für CIRCLE RC

der Anzeigezeigezeitraum der Produktdaten werden in die Seitenleiste integriert, dann als entsprechender Menüpunkt(4). Die Auswahl der Maschinen und Ähnlichem bleibt erhalten(2). Die Anordnung der eigentlichen Menüauswahl bleibt ebenfalls unverändert, Veränderungen im Funktionsumfang sind nicht berücksichtigt(5). In Abschnitt Drei wird der Anzeigezeitraum aus der Kopfleiste integriert, die Anzeige der jeweiligen Schicht bleibt erhalten. Über diesen Abschnitt wird auch der Kalender aufgerufen. Den Rest der Anwendung macht das Funktionsfenster aus(6), in dem wie gewohnt alle Funktionen angezeigt werden.

Das ähnelnde Layout soll dem Benutzer den Einstieg erleichtern. Ein aktuelleres Design wird mit den gegebenen Mitteln realisiert. Eine Komfortfunktion wäre beispielsweise ein klappbares Menü, so dass die Unterpunkte erst durch Klicken des Menüpunktes sichtbar „ausgefahren“ werden. Für diese Funktion ist ausreichend Platz in der Seitenleiste.

Die zweite Variante beinhaltet weitreichendere Änderungen am Layout als Kandidat Eins. Lediglich die Titelleiste stimmt mit Layout Eins überein(1). Wie zu

erkennen ist, fällt die Seitenleiste komplett weg. Sie wird durch eine umfangreiche Kopfleiste ersetzt, die alle Funktionen enthält(2). Sie ist in Themen gegliedert, ähnlich der Menüpunkte im CIRCLE Analyzer. Als Vorbild könnte das Ribbon Interface der Microsoft Office Suite dienen. Alle Funktionen werden in einer Leiste geführt ohne Dropdown-Listen. Ferner wird eine Tableiste für aufgerufene Funktionen eingeführt(3). So kann schnell zwischen Funktionen gewechselt werden. Das Funktionsfenster ist jetzt auf die volle Breite ausgedehnt und bietet ausreichend Platz für die Anzeige der Funktionen. Natürlich wird auch hier ein modernes Design verwendet.

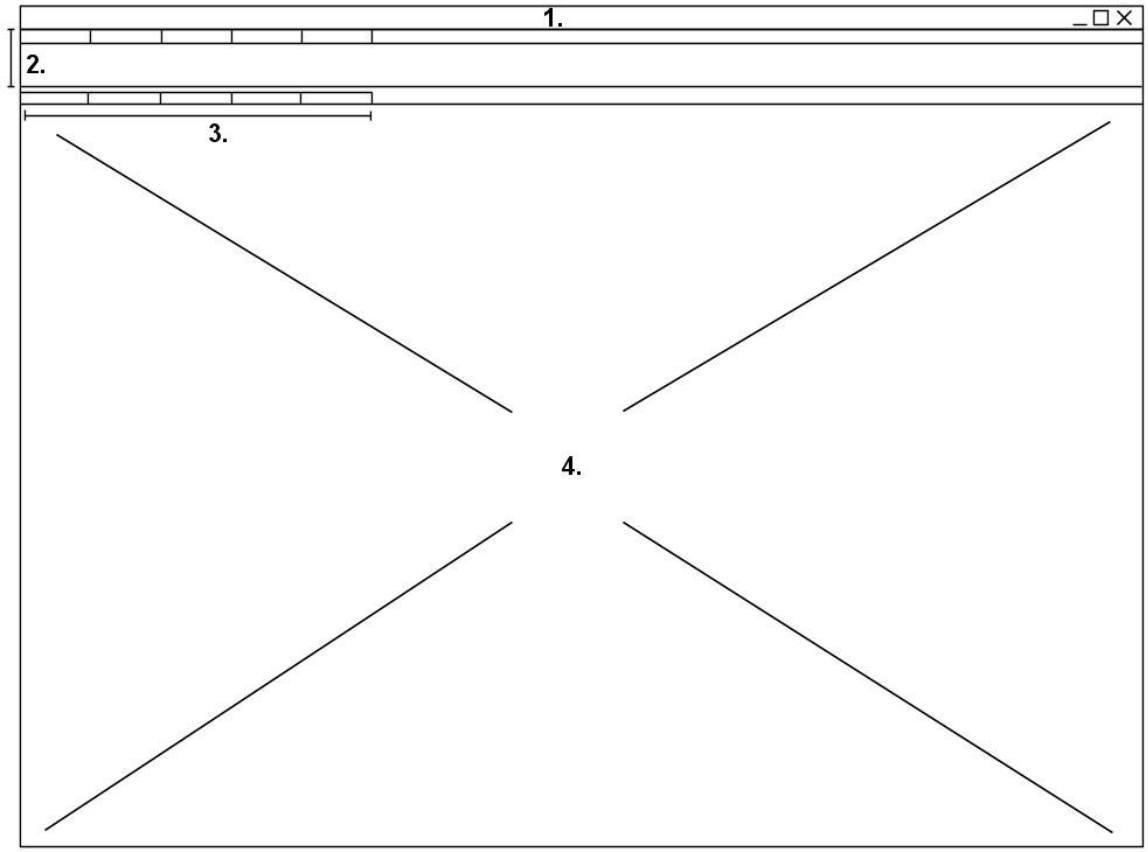

Abbildung 16 – Entwurf eines Layouts für den CIRCLE RC

Fern von diesen Layouts könnte man für beide Varianten eine Favoritenliste einführen. Dort werden die am häufigsten genutzten Funktionen gelistet.
Wie die Standardoberflächen mit Eclipse und SWT aussehen, wurde bereits in Kapitel 3.3.4 erwähnt.

4.3. Die OSGi Spezifikation

OSGi steht für Open Services Gateway Initiative. Dahinter steht die OSGi Alliance, in der Firmen wie IBM und Nokia vertreten sind. Es ist eine reine Spezifikation für Java, die auf den Möglichkeiten der Java Standard Edition aufbaut. Da eine Java Virtual Machine vorausgesetzt wird, kann dieser Standard überall dort genutzt werden, wo eine JavaVM einsetzbar ist. Die aktuelle Version 4.2 stammt von September 2009, die erste Veröffentlich war im Jahr 2000. Ursprünglich wurde OSGi für den Bereich der eingebetteten Anwendungen entwickelt. Darunter fallen Mobile Device Anwendungen oder auch Residential Internet Gateway Systeme. Aufgrund der gegebenen Merkmale findet es auch in der allgemeinen Java-Entwicklung Anklang. Die Spezifikation wurde zum Java Standard erhoben und im JSR 291 beschrieben. Um praktischen Nutzen daraus ziehen zu können, muss es eine Implementierungen dessen geben. Eclipse Equinox ist als Beispiel zu nennen. Weitere Implementierungen sind Apache Felix oder auch SwingOSGi.

4.3.1. Theoretisches

Wie mehrfach erwähnt, bietet die OSGi Spezifikation die Möglichkeit Java Anwendungen modularer zu gestalten. Um eine Realisierung zu gewährleisten, werden verschiedene Techniken genutzt, eine davon ist das Komponentenmodell. Dazu gehört die Verwendung von Bundles. Das sind separate Module, rein technisch gesehen sind es jar[12] Dateien. Im Kontext des OSGi Standards werden sie aber als Bundles bezeichnet.
Mit dem Konzept der Dynamik wird eine möglichst lose Kopplung erreicht. Dazu bedient man sich einer Service Registry und Bundles Services. Diese Stellen die Schnittstellen, über die eine Kommunikation mit anderen Bundles erfolgt, dar. Die Service Registry ermöglicht das An- und Abmelden von Services zur Laufzeit des Programms. So können neue Bundles jederzeit in das Programm eingebunden werden, zu diesem Thema gibt es spätere weitere Details.

[12] JAR Dateien – javaspezifische gezippte Container, die sämtliche Ressourcen einer Anwendung enthalten

Nun muss aber auch erwähnt werden, dass das Konzept der Modularisierung nicht neu ist. Im Bereich der Java Anwendungen wird eine Aufteilung durch Packages gewährleistet. Bundles sind als eine weitere Schicht oberhalb der Packages anzusehen. Zu Beginn soll aber der allgemeine Aufbau des OSGi Standards betrachtet werden. Das lässt sich an einem Schichtmodell verdeutlichen.

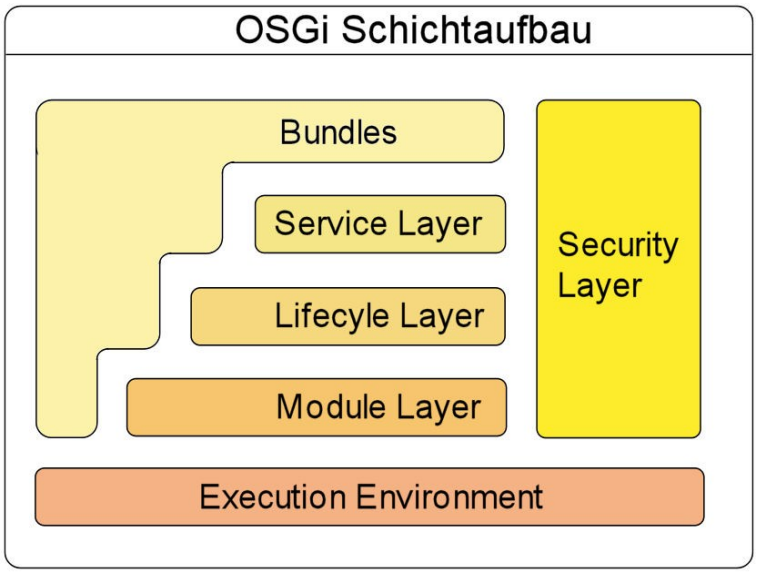

Abbildung 17 – OSGi Schichtaufbau

Die unterste Ebene bildet das Execution Environment. Eine solche Laufzeitumgebung ist die Implementierung des OSGI Standards. So ist Eclipse Equinox eine solche Implementierung. Unterhalb von Eclipse Equinox liegt die Java Runtime. So kann auch bestimmt werden, welche Mindestversion auf dem Hostsystem vorhanden sein muss. Den Abschluss bildet das Betriebssystem. Eine Beschränkung dessen ist nur durch die Unterstützung der Java Runtime gegeben. Im Module Layer sind alle benötigten Ressourcen beinhaltet, die für den Aufbau eines Bundle genutzt werden. Darunter fallen neben den Java Klassen auch Medien wie Videos oder Audiodaten. Der Lifecycle Layer beschreibt die Dynamik der Bundles. Es wird festgelegt wie die Module zur Laufzeit eines Programms de- und installiert, gestartet und gestoppt werden. Der Service Layer verwaltet die Funktionalitäten, die mit den Services der Bundles einhergeht. Die Services sind im Grunde reine Java Objekt, auch in einiger Literatur als POJO[13] bezeichnet. Hier ist die ServiceRegistry zu nennen, die die

[13] Pojo – Plain Old Java Objects

Verwaltung der Services übernimmt. Der Service Layer spielt eine wichtige Rolle, denn Bundles kommunizieren nur über ihre API. Der Security Layer basiert auf der Java2Security. Dieser Layer durchzieht alle Ebenen eines Bundles. Es erfolgen Regelungen zum Signieren von Bundles und die Verteilung von Berechtigungen an Bundles. Es kann beispielsweise bestimmt werden, welche Bundles Zugriff auf die Services eines anderen Bundles hat. Es ist auch möglich Restriktionen im Zugriff auf externe Klassen und Ressourcen zu verteilen.
Wie erwähnt sind Bundles rein technisch .jar Dateien. Neben gewohnten Klassen und Packages können auch eingebettete .jar Dateien enthalten sein. Weiterhin werden benötigte Ressourcen und beschreibende Daten mit eingebunden. Letztere sind wichtig um Abhängigkeiten zu anderen Bundles, konkret ihrer Services, zu definieren. Dazu dient die Manifest.mf. Es stehen zwei Möglichkeiten zur Auswahl, entweder werden nur die benötigten Klassen anderer Bundles festgelegt, analog dazu verhält es sich mit den Klassen und Packages, die für andere Module zur Verfügung gestellt werden. Die zweite Möglichkeit ist die Angabe ganzer Bundles und deren Klassen. Die erste Möglichkeit bietet den Vorteil, dass man sich nicht von bestimmten Bundles abhängig macht, denn es ist durchaus möglich, dass andere Bundles auch die benötigten Klassen bereit stellen. Im Rahmen der OSGi Spezifikation benötigt ein Bundle keine Informationen woher es die zu importierenden Packages kommen. So wird eine möglichst lose Kopplung der Bundles erreicht. Um eine eindeutige Identifikation eines Bundles vorzunehmen werden erhält es eine Versionsnummer und einen sogenannten „*symbolic name*“, beide müssen explizit in der Manifest Datei angegeben werden. Weiterhin erhält ein Modul eine einzigartige ID.

Bundles besitzen einen Lebenszyklus. Sie können sechs Zustände erhalten.

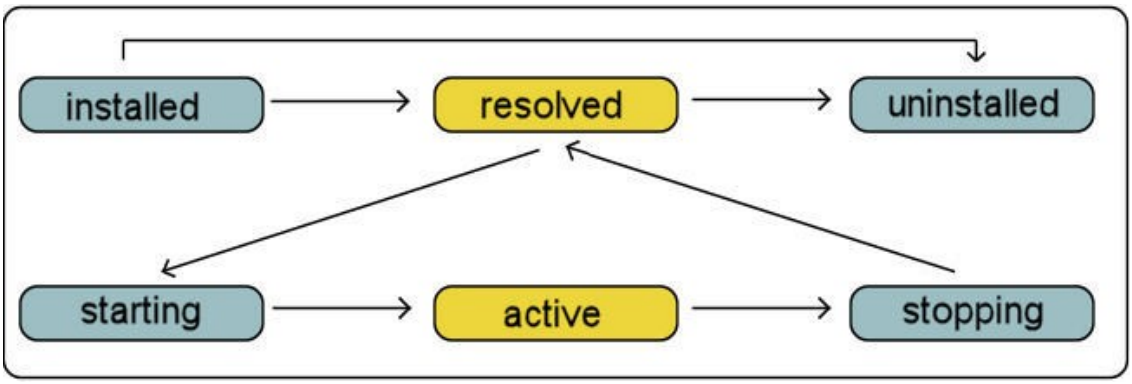

Abbildung 18 – Bundle Lifecycle

Zu Beginn gilt ein Bundle als „*installed*“. Es wird innerhalb der Anwendung bekannt gemacht und in den Bundle Cache abgelegt. Die Bekanntmachung erfolgt, in dem eine Liste vorhandener Bundles erstellt wird und dies auf ein gültiges Format und gültige Metadaten geprüft wurden. In diesem Zusammenhang wird auch eine eindeutige ID vergeben. Ein Bundle tritt in den Status „*resolved*“ ein, wenn alle Abhängigkeiten zu anderen Bundles geklärt sind. Dabei werden die zu im- und exportierenden Klassen angegeben, sowie die Bereitstellung eventuell vorhandener Services ausgeführt. Die Bereitstellung erfolgt an der sogenannten ServiceRegistry. Stehen in der gesamten Anwendung mehrere Packages mit gleichen Funktionen zur Verfügung, wird im ersten Schritt die Package Versionsnummer vergleichen. Reicht das nicht aus, erfolgt ein Vergleich der Bundle Versionen. Ist immer noch keine eindeutige Identifikation gegeben, wird das Bundle mit der kleinsten ID gewählt. Im „*resolved*“ Status gilt ein Modul als geladen und kann genutzt werden. Es gibt zwei Ladestrategien. Die Erste löst die Abhängigkeiten zum Programmstart auf, die zweite übernimmt dies erst beim erstmaligen Gebrauch eines Bundles. Diese Art der Aktivierung ist ressourcenschonender. Ein Bundle kann ebenfalls wieder deinstalliert werden. Wenn das Modul genutzt werden soll, muss es gestartet werden. Dazu wird die Funktion BundleActivator verwendet, welche im BundleContext eingetragen wird. In diesem Moment erhält es den Status „*starting*“. Ist diese Phase überwunden, gilt es als „*active*“. Wird ein Bundle gestoppt, erfolgt eine Rückführung in den Zustand „*resolved*“. Treten beim Starten oder Stoppen Probleme auf wird es in den Zustand „*resolved*“ gebracht oder bleibt als „*active*“ bestehen. Wird ein Bundle deinstalliert und andere Bundles nutzen die exportierten Packages, bleiben diese bis zu einem *refresh* erhalten. Es gewährt die einwandfreie Funktion anderer abhängiger Bundles. Ein Bundle befindet sich im Status „*uninstalled*“ wenn es selbst nicht aktiv ist und keinen Service oder Package exportiert hat. Das Hilfsmittel der Versionierung ist von Bedeutung für den Updateprozess. Wird ein Bundle aktualisiert, so wird dieses in den Ursprungszustand versetzt. Ein gestartetes Bundle wird gestoppt. Es erfolgt das Update und ein anschließendes neues Einlesen und Auflösen der Abhängigkeiten, sowie das Starten. Auf diese Weise bleibt die ID erhalten. Falls Abhängigkeiten zu anderen Modulen bestehen, muss entsprechend darauf reagiert werden. Die Spezifikation sieht dafür *BundleListener* vor. Sie reagieren

auf *BundleEvents* wie der Änderung des Bundlestatus. Die Veränderung muss aber vom aktiven Bundle selbst übermittelt werden. So können andere Bundles auf die aktualisierten Bundles zugreifen. Ohne eine Reaktion würden die Module die ursprünglich exportierten Packages und Klassen weiternutzen.
Es gibt noch eine weitere Art von Bundles. Sie heißen Fragments und sind im Grunde wie Bundles aufgebaut, mit dem Unterschied, dass diese nicht separat funktionieren. Fragmente besitzen einen Host, an welchen sie gebunden sind. Bundle Fragemente erweitern ein vorhandenes Bundle. Dabei wird zur Laufzeit der Bundle Manifest Header mit Fragment Manifest Header verbunden und alle Abhängigkeiten aufgelöst.
Treten Fehler im Header des Fragmentes auf, werden diese nicht eingebunden, das Host Bundle läuft dann im normalen Zustand weiter.
Neben Packages und Klassen stellen Bundles auch Services bereit. Diese bieten ebenfalls eine hohe Dynamik und erhalten eine ähnliche Verwaltung. Zentrales Objekt ist die sogenannte ServiceRegistry. Bereitgestellte Services werden an ihr registriert und andere Bundles erhalten somit Zugriff auf diese. Ein Service ist eine Instanz eines Java Objekts. Über die Klasse *BundleContext* wird eine Registrierung eines Services an der ServiceRegistry vorgenommen. Zu Registrierung wird ein Service Interface genutzt. Das ServiceInterface unterstützt das Prinzip der losen Kopplung. Es verlangt bestimmte Voraussetzungen, die ein Service erfüllen muss. Wird ein ServiceInterface genutzt, wird vor der eigentlichen Nutzung des Service eine *ServiceReference* erfragt. Grund für die *ServiceReference* ist die Möglichkeit über ein *Service Interface* mehrere Services registrieren zu können. Bundles erfragen *die ServiceReference* um einen Service nutzen zu können. Sind mehrere passende Services vorhanden, wird derjenige mit der kleinsten Service ID verwendet. Sollten keine passenden Services vorhanden sein, wird eine 0 zurückgeliefert. Solange die Anforderungen des Interface erfüllt sind, kann ein Service registriert werden und genutzt werden. Bei der Registrierung erhalten die Services ID Nummern.
Services können ebenso wie die Bundles während des Betriebs registriert oder auch deregistriert werden. Um auf diese Gegebenheit zu reagieren, gibt es die Objekte *ServiceListener* und *ServiceTracker*. Sie bieten Events um auf das De- und Registrieren zu reagieren und müssen im *BundleContext* angegeben sein. Zudem können Services nur genutzt werden, wenn das exportierende Bundle

den Status „*starting*“, „*active*“ oder „*stopping*“ einnimmt, also anders als bei exportierten Packages, dessen Klassen und Packages auch im „*resolved*“ Status genutzt werden können. Services hingegen können genutzt werden, sobald alle Abhängigkeiten der Bundles geklärt sind.

Bei großen Projekten kann das OSGi Framework zu hoher Komplexität führen, langen Startzeiten und einem hohem Speicherbedarf. Die Komplexität ergibt sich wie folgt. Ein Bundle benötigt eine bestimmte Anzahl von anderen Bundles beziehungsweise deren exportierte Klassen. Gleiches gilt für Services, die über die ServiceRegistry angeboten werden. Daraus resultiert eine Überwachung dieser Bundles und Services mittels *BundleListener* oder *ServiceListener* und *ServiceTracker*.

Durch das Registrieren aller Services beim Start der Bundles kann es zu langen Startzeiten und einem hohen Speicherbedarf kommen. Es gibt keine Berücksichtigung ob die Services zum Start gebraucht werden oder nicht.

Um diesen Problemen entgegenzuwirken, enthält die OSGi Spezifikation die Declarative Services Spezifikation. Das Declarative Service Component Model soll den Umgang mit Services vereinfachen. Ein *declarative Service* wird als Service Components bezeichnet. Es besteht aus einer XML Beschreibung, der *Component Description*, und einem Objekt, der *Component Instance*. Die *Component Description* enthält alle Informationen über das Objekt, wie beispielweise auch das ServiceInterface. In der Manifest Datei wird eine Referenz auf die *Component Description* angelegt. Beim Starten des Bundles wird die XML Datei ausgewertet und eine *Service Component* erzeugt. Diese stellen dann die Services zur Verfügung.

Eine weitere Lösung sind *Delayed Components*. Sie können den Start von Bundles und Services verzögern, was sich positiv auf die Startzeit und den Speicherverbauch einer Anwendung auswirken kann. Ein Service wird nicht sofort beim Start erzeugt. Stattdessen wird ein Platzhalter erstellt und an die ServiceRegistry angemeldet. Erst wenn der Service das erste Mal genutzt wird der Service erstellt.

4.3.2. Nutzen für CIRCLE

Der CIRCLE Rich Client zieht folgende Vorteile aus der Implementierung des OSGi Standards in Eclipse. Den größten Nutzen bietet das Komponentenmodell, mit dem man versucht eine gewisse Struktur und möglichst unabhängige Komponenten zu gewährleisten und den CIRCLE Rich Client modular zu gestalten. Weitere Vorteile sind die Versionierung und das Update Management.

4.4. Message Broker – ActiveMQ

ActiveMQ ist ein Message Broker, also ein Nachrichtenverwalter. Mit dieser Anwendung wird eine zentrale einheitliche Sammelstelle für alle Nachrichten in einem System geschaffen. Es ist als Middleware in einem Client-Server System anzusehen. Großer Vorteil ist die Kommunikation verschiedener systemfremder Anwendungen. Die Umsetzung der JMS Standards gewährt eine lose Koppelung der Systeme und ermöglicht eine einfachere Austauschbarkeit dieser.

4.4.1. Theoretisches

ActiveMQ ist ein Projekt der Apache Foundation. Somit ist es als Open Source Anwendung zu erhalten. Ziel ist der Datenaustausch verschiedenster Systeme. Dazu bedient sich ActiveMQ des Java Message Service Standards und verschiedener Nachrichtenprotokolle. Neben dem JMS Standard werden Nachrichtenprotokolle wie STOMP und OpenWire unterstützt. Dadurch ergibt sich eine große Anzahl an unterstützten Skript- und Programmiersprachen. Darunter fallen C, Ruby, Perl, Python, PHP und ActionScript um nur einige zu nennen.

ActiveMQ ist in Java geschrieben und benötigt die Java Runtime in der minimalen Version von 1.4. Der Nachrichtendienst implementiert einen vollwertigen JMS Provider. Das ist die Implementierung des JMS Standards. JMS wird in der aktuellsten Version 1.1 unterstützt. Der JMS Standard ermöglicht die asynchrone Kommunikation zwischen verschiedenen Softwaresystemen. Die Nachrichten werden zwischen zwei Systemen verteilt, wichtig ist nur, dass ein einheitliches Nachrichtenformat vorgegeben ist, um die Kommunikation zwischen

den verschiedenen Systemen zu gewährleisten. Zu den weiteren Features gehören neben der Unterstützung der, die dauerhafte Speicherung von empfangenen Nachrichten. Sollte es zu Problemen bei den Clients oder bei ActiveMQ selbst kommen, können die Nachrichten wieder aufgenommen werden.

Eine JMS Nachricht hat einen dreigeteilten Aufbau. An oberster Stelle befindet sich der Nachrichtenkopf, auch Header genannt. Er wird für die Identifikation und zur Übertragung genutzt. Nachrichten können auch mit Eigenschaften versehen werden, den Properties. Den abschließenden Teil bildet der Nachrichtenrumpf. Dieser enthält die eigentlichen Daten, die übermittelt werden sollen.

Durch den Java Message Service werden zwei Übertragungsmodelle möglich. Das ist zum einen eine Point-to-Point Übertragung, zum Anderen ist eine Publish/Subscribe Nachrichtenverteilung möglich. Beide Modelle werden vorgestellt.

Bei einer Point-to-Point Übertragung wird eine gesendete Nachricht nur zwischen zwei Clients ausgetauscht, ein Sender und ein Empfänger. Der sendende Client verschickt eine Nachricht an den JMS Provider. Für jeden Client wird eine eigene Warteschlange(Queue) erstellt. An welchen Client diese Nachricht gerichtet ist, wird im Header der Nachricht vermerkt. Die gesendeten Nachrichten werden in der Warteschlange aufgereiht bis der empfangende Client die Nachrichten erhalten hat, im JMS Kontext auch konsumieren genannt, oder aber die Nachrichten werden nach einer bestimmten Zeit wieder gelöscht. Bei Verwendung des JMS Standards wird der Sender einer Nachricht nicht blockiert, während dieser auf eine Antwort wartet. Somit können weitere Nachrichten geschickt werden. Das heißt wenn der Empfänger die Nachricht noch nicht erhalten hat, kann trotzdem eine weitere Nachricht vom Sender in die Warteschlange eingereiht werden. Sie werden nach dem First in First out Prinzip verschickt. Die Nachrichten, die zuerst in die Warteschlange eingereiht wurden, werden auch in der gleichen Reihenfolge verschickt. Hat der Empfänger die Nachricht erhalten, schickt dieser ein Acknowledge an die Queue. Damit erhält der JMS Provider einen Bericht, dass Nachricht empfangen wurde und diese gelöscht werden kann.

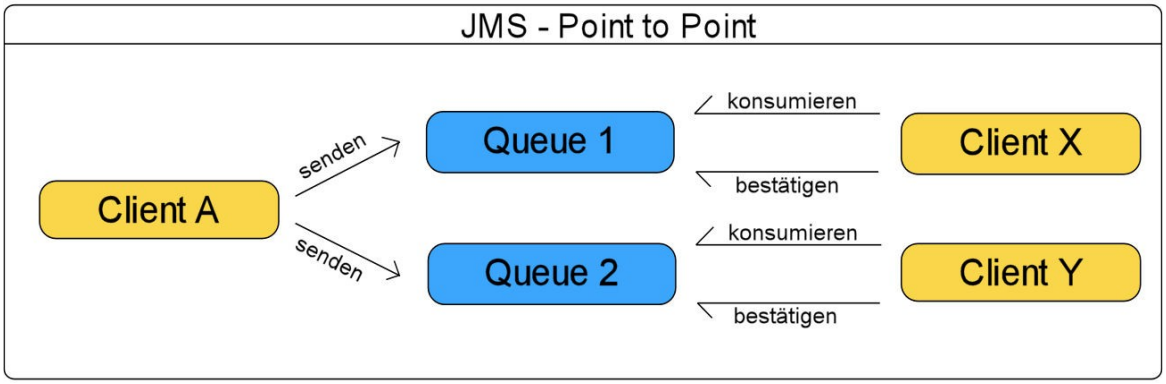

Abbildung 19 – Nachrichtenaustausch im Point to Point „Modus“

Um eine Kommunikation zwischen Client und JMS Provider aufzubauen, wird ein ConnectionFactory Objekt erzeugt im Client erzeugt. Im weiteren Verlauf wird über das Connection Objekt eine Verbindung zum JMS Provider aufgebaut, also zum Beispiel ActiveMQ. Nach dem Verbindungsaufbau wird eine Session erzeugt. Nun können Nachrichten gesendet werden. Diese Art eine Verbindung aufzubauen, trifft bei beiden Übertragungsmodellen zu, jedoch werden unterschiedliche Klassen genutzt.

Der Publish/Subscribe Modus baut sich wie folgt auf. Es gibt ein Client der eine Nachricht an ein sogenanntes Topic sendet. Der Unterschied zur Point-to-Point Verbindung besteht in dem Veröffentlichen der Nachricht für beliebig viele empfangende Clients. Sie registrieren sich an diesem Topic. Es ist auch eine Warteschlange, in der die Daten abgelegt werden. Die registrierten Clients erhalten erst ab dem Zeitpunkt ihrer Verbindung zum Provider gesendete Nachrichten. Es besteht die Möglichkeit, dass Daten erhalten bleiben. So können Clients in der Verbindung unterbrochen werden und bei einem Wiederverbinden die in der Zwischenzeit gesendeten Nachrichten erhalten. Diese Option ist jedoch mit einem Mehraufwand verbunden, da die Daten extra vorgehalten und zusätzliche Daten zum Client abgelegt werden müssen. Als Ausgleich gehen aber keine Nachrichten verloren.

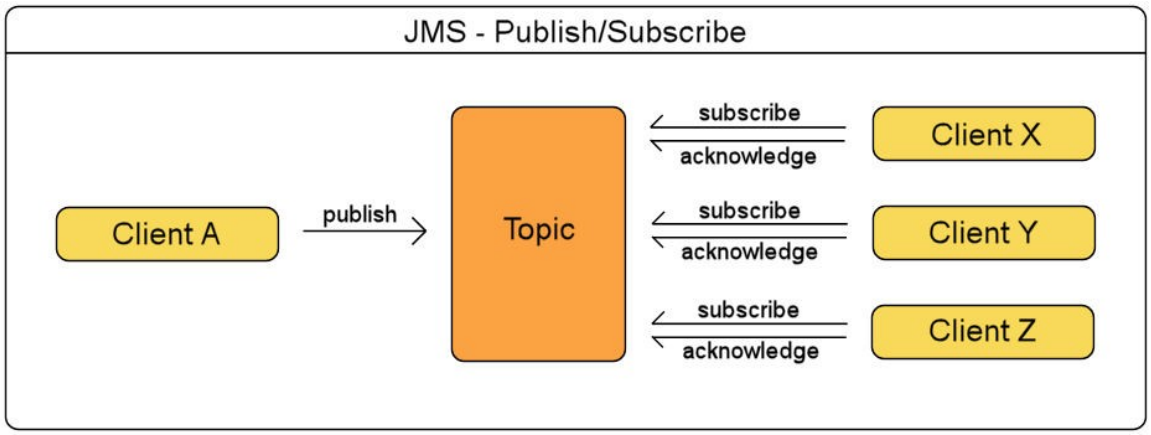

Abbildung 20 – Nachrichtenaustausch im Publish/Subscribe „Modus“

Der Rich Client muss einen MessageService einbinden um mit ActiveMQ kommunizieren zu können. Es fehlt nun aber noch die Anbindung an den CIRCLE Server um Daten aus der Datenbank zu erhalten. Auch wenn ActiveMQ in Java geschrieben ist, bietet ein Subprojekt eine Lösung. Sie nennt sich ActiveMQ-CPP und verwendet einen dem JMS ähnlichen Service, den CMS. Mit CMS wird versucht ein einheitliches Nachrichtenmodell zwischen Java Applikation und C/C++ Applikation zu gewährleisten. Alternativ könnte mit einem der unterstützten Protokolle ein Datenaustausch zwischen dem CIRCLE Server und ActiveMQ stattfinden.

5. Zusammenfassung

In dieser Bachelorarbeit wurden das Auswertungs- und Überwachungsprogramm CIRCLE auf die Möglichkeiten der Verwendung von Rich Client und Rich Internet Applikation geprüft. Das CIRCLE System soll nicht nur mit Hilfe der Rich Clients und Rich Internet Application erweitert werden. So soll die Kommunikation zwischen den Anwendungen mit Hilfe von ActiveMQ zentralisiert werden. Hierfür wurde eine Analyse des vorhandenen Systems vorgenommen. Im Speziellen wurde das Visualisierungsprogramm des Systems, der CIRCLE Analyzer, begutachtet. Denn hier ist ein Ansatzpunkt für die Rich Client und Rich Internet Application gegeben. Es wurden dazu JavaFX, Microsoft Silverlight, Adobe Flex, die EclipseRCP und das RAP Framework für einen möglichen Einsatz im CIRCLE System ausgewählt. Die ausgewählten Technologien wurden vorgestellt und verglichen. Abschließend wurde ein Kandidat für die Weiterverwendung

ausgewählt. Die endgültige Auswahl fiel auf die EclipseRCP, somit wird ein neuer Desktop Client des CIRCLE Analyzers auf Basis von Java erstellt. Ausschlaggebend waren die zusätzlich zu nutzende Middleware ActiveMQ und die OSGi Spezifikation, welche in Eclipse implementiert ist. Beide nutzen Java beziehungsweise bauen darauf auf, so ist die EclipseRCP eine passende Ergänzung. Für das neue System wurden zwei Layout erstellt und mögliche Neuerungen im Vergleich zum originalen CIRCLE Analyzer aufgeführt. ActiveMQ und OSGi finden auch im neuen System Verwendung, deshalb wurden diese ebenfalls begutachtet und vorgestellt.

Quellenverzeichnis

Bücher/Zeitschriften:

[1] Clausohm, Michael – CIRCLE Analyzer

[2] Wütherich, Gerd und Nils Hartmann, Bernd Kolb und Matthias Lübken - Die OSGi Service Platform, dpunkt Verlag, 2008

[3] Böck, Heiko – Netbeans Platform 6 – Rich Client Entwicklung mit Java Galileo.Press, 2008

[4] Tune, Timothy: Rich Clients, Java Magazin 11/08, entwickler.press, 2008

[5] Seeberger, Heiko: OSGi in kleinen Dosen Teil 1, Java-Magazin 12/08, entwickler.press, 2008

[6] Seeberger, Heiko: OSGi in kleinen Dosen Teil 2, Java-Magazin 01/09, entwickler.press, 2009

[7] Seeberger, Heiko: OSGi in kleinen Dosen Teil 3, Java-Magazin 02/09, entwickler.press, 2009

[8] Seeberger, Heiko: OSGi in kleinen Dosen Teil 4, Java-Magazin 03/09 entwickler.press, 2009

[9] Röwekamp, Lars: JavaFX, Java-Magazin 12/08, entwickler.press, 2008

[10] Müller, Björn: Rich-Client-Architekturen, Java-Magazin 12/08, entwickler.press, 2008

[11] Sternberg, Ralf und Benjamin Muskalla: Rich Ajax Platform, Java-Magazin 02/09, entwickler.press, 2009

Internet:

[12] Eclipse Dokumentation
www.eclipse.org – letzter Zugriff: 01.10.2009

[13] OSGi Alliance Specification
www.OSGi.org – letzter Zugriff: 01.10.2009

[14] Adobe Flex Builder
http://www.adobe.com/products/flex/features/flex_builder/ – letzter Zugriff: 01.10.2009

[15] Adobe Flex Einblick
http://fleksray.org/ – letzter Zugriff: 01.10.2009

[16] Definition Rich Client – kioskea.net
http://de.kioskea.net/contents/cs/client-riche.php3 – letzter Zugriff: 01.10.2009

[17] Definition Rich Client – wikipedia.de
http://de.wikipedia.org/w/index.php?title=Fat_Client&oldid=64151522 – letzter Zugriff: 01.10.2009

[18] RAP – Introduction
http://www.eclipse.org/rap/introduction.php – letzter Zugriff: 01.10.2009

[19] SWT: The Standard Widget Toolkit
http://www.eclipse.org/swt/ – letzter Zugriff: 01.10.2009

[20] RIA Flex Anwendung Picnik
http://www.picnik.com/app#/home – letzter Zugriff: 01.10.2009

[21] Featureliste des Java 6 Update 10
https://jdk6.dev.Java.net/plugin2/ – letzter Zugriff: 01.10.2009

[22] Reinhold, Mark – Neuerungen in Java 7
http://hamletdarcy.blogspot.com/2008/12/java-7-update-from-mark-reinhold-at.html – letzter Zugriff: 01.10.2009

[23] RAP Release
http://www.golem.de/0710/55379.html – letzter Zugriff: 01.10.2009

[24] Adobe Flex Release
http://opensource.adobe.com/wiki/display/flexsdk/Download+Flex+3 – letzter Zugriff 01.10.2009

[25] Rich Internet Application and AJAX
http://www.javalobby.org/articles/ajax-ria-overview/ – letzter Zugriff: 02.10.2009

[26] Eclipse Foundation – Downloads
http://www.eclipse.org/downloads/packages/release/ganymede/sr2 – letzter Zugriff: 03.10.2009

[27] JavaFX - API Documentation
http://java.sun.com/javafx/1.2/docs/api/index.html# – letzter Zugriff: 03.10.2009

[28] JavaFX - Release
http://java.sun.com/javafx/1/reference/releasenotes/javafx-sdk-release-notes-1-2-1.html – letzter Zugriff: 01.10.2009

[29] JavaFX - Features
http://www.sun.com/software/javafx/features.xml – letzter Zugriff: 03.10.2009

[30] Adobe Flex
http://www.adobe.com/de/products/flex/overview – letzter Zugriff: 03.10.2009

[31] SWT
http://www.eclipse.org/swt/ – letzter Zugriff: 03.10.2009

[32] Seeberger, Heiko – Erste Schritte mit OSGi
http://it-republik.de/jaxenter/artikel/Erste-Schritte-mit-OSGi-2077.html – letzter Zugriff: 03.10.2009

[33] Zörner, Stefan – Applikationsdesign im Zeitalter von OSGi
http://it-republik.de/jaxenter/artikel/Applikationsdesign-im-Zeitalter-von-OSGi-1898.html – letzter Zugriff: 03.10.2009

[34] Riastats
www.riastats.com – letzter Zugriff: 03.10.2009

[35] Silverlight 3 Anwendung
http://videos.visitmix.com/MIX09/T45F – letzter Zugriff: 03.10.2009

[36] Silverlight Review 2007
http://www.nsaneblog.com/software/260/microsoft-silverlight-review/ – letzter Zugriff: 03.10.2009

[37] Pfeil, Christian – Adobe AIR
http://www.christianpfeil.com/warum-adobe-air/ – letzter Zugriff 03.10.2009

[38] Wikipedia – Rich Internet Application
http://de.wikipedia.org/w/index.php?title=Rich_Internet_Application&oldid=64401147 – letzter Zugriff 03.10.2009

[39] What is Rich Internet Application?
http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci1273937,00.html – letzter Zugriff 03.10.2009

[40] Rich Internet Application – Ein Überblick
http://createordie.de/cod/artikel/Rich-Internet-Applications-&ndash%3B-Ein-Ueberblick-1791.html – letzter Zugriff 03.10.2009

[41] Ebert, Ralf – Informationen zum Eclipse RCP Buch
http://www.ralfebert.de/rcpbuch/info/ – letzter Zugriff 03.10.2009

[42] Burnette, Ed – EclipseZone – What is a Rich Client? - 2005
http://www.eclipsezone.com/eclipse/forums/t59350.html – letzter Zugriff 03.10.2009

[43] Java Message Service Specification
http://java.sun.com/products/jms/docs.html – letzter Zugriff 03.10.2009

[44] Computerwoche.de – Silverlight 2 schlägt Brücke zum Rich Client
http://www.computerwoche.de/software/software-infrastruktur/1869877/ – letzter Zugriff 03.10.2009

[45] Sun.com – Swing and JDK 7
http://blogs.sun.com/theplanetarium/entry/the_future_of_swing – letzter Zugriff 03.10.2009

Tabellenverzeichnis

Abbildungsverzeichnis

Anhang

Digitale Kopie der Bachelorarbeit auf CD