Usability Engineering mit Eclipse RCP

Julia Maly
User Interface Design GmbH
Martin-Luther-Straße 57-59
71636 Ludwigsburg
julia.maly@uid.com

Henning Brau
German UPA
Postfach 80 06 46
70506 Stuttgart
hb@germanupa.de

Bernd Watzal
German UPA
Postfach 80 06 46
70506 Stuttgart

Abstract

Eclipse RCP ist ein Open-Source-Framework, das sich zunehmend als Entwicklungsumgebung - vor allem für SOA-Vorhaben - etabliert. Durch das spezifische Perspektiven-Konzept, das die parallele Bearbeitung verschiedener Aufgaben ermöglicht, ergeben sich neue Chancen für die Gestaltung system-übergreifender, aufgabenorientierter Anwendungen. Doch diese Flexibilität hat ihre Herausforderungen für das User-centered Design: In diesem Vortrag wird die Bedienphilosophie des Perspektivenkonzepts von Eclipse RCP erläutert und die sich ergebenden Implikationen für das Interface Design und die Arbeit als Usability Professional diskutiert.

Keywords

Eclipse, RCP, Usability Engineering, Perspektiven, Views

1.0 Einleitung

Eclipse RCP als Entwicklungsplattform wird immer populärer. In diesem Zusammenhang werden auch Usability Professionals immer häufiger mit der Gestaltung von User Interface Konzepten für und mit Eclipse RCP konfrontiert.

Der im nachfolgenden geschilderte Erfahrungsbericht aus der Entwicklung einer Desktop-Anwendung im Industrieumfeld mittels Eclipse RCP soll einen subjektiven Eindruck in die Arbeit mit Eclipse als Usability Professional geben und zur Diskussion über die Stärken und Schwächen anregen.

2.0 Die Eclipse-Plattform

Eclipse ist ein Open-Source-Framework zur Entwicklung von Software nahezu aller Art, das sich zunehmend als Entwicklungsumgebung - vor allem für SOA-Vorhaben - etabliert.

Das Eclipse-Projekt wurde ursprünglich von IBM 2001 ins Leben gerufen, ging jedoch 2004 in die Verantwortung der hierfür gegründeten Eclipse Foundation (www.eclipse.org) über, um eine transparentere Entwicklung von Eclipse als Open-Source-Plattform zu gewährleisten (Eclipsepedia 2009).

Der ursprüngliche Verwendungszweck der Eclipse-Plattform in ihrer Gesamtheit war dabei die Umsetzung von integrierten Entwicklungsumgebungen für die Softwareentwicklung – so genannten IDEs (integrated development environments). Mit Eclipse 3.0 wurde schließlich eine Untermenge der Eclipse Komponenten veröffentlicht: die Eclipse Rich Client Platform (RCP). Eclipse RCP bündelt dabei alle Komponenten der Eclipse-Plattform, die für die Entwicklung von Applikationen außerhalb von IDEs relevant sind (des Rivieres & Beaton, 2006). Mit diesem Schritt wurde Eclipse mehr und mehr für die Entwicklung von Softwareanwendungen in nahezu allen Branchen und Industriezweigen interessant und ist es bis heute geblieben (Eclipsepedia 2009).

2.1 Die Eclipse UI-Philosophie

Für Erscheinungsbild und Aufbau der Benutzungsoberfläche einer Eclipse RCP-Anwendung sind vor allem vier Begriffe von Relevanz:

- Workbench
- Perspektive
- Views
- Editoren

Die so genannte Workbench bildet den Applikationsrahmen, in dem sich die Inhalte der jeweiligen Anwendung ansiedeln können (Ebert 2009). Prinzipiell ist die Workbench daher mit dem Anwendungsfenster gleichzusetzen.

Innerhalb der Workbench sorgen die so genannten Perspektiven für Struktur im Aufbau der Oberfläche. Eine Perspektive ist in Eclipse RCP mit einer Seite in einem Buch zu vergleichen. Das Buch (= die Anwendung) kann dabei beliebige viele Seiten (= Perspektiven) haben. Jede Seite kann unterschiedliche Inhalte und Layouts aufweisen, je nach Thema und hierfür angemessener Struktur. Der Anwender kann in der Regel frei entscheiden, welche der Perspektiven er ‚aufschlägt'. Es ist jedoch immer nur eine Perspektive auf einmal in der Workbench anzeigbar. Das ‚Umblättern' also das Wechseln der Perspektive, erfolgt über die Perspektivenleiste (Eclipsepedia 2009).

Das Perspektivenkonzept hat seine Wurzeln in der Verwendung von Eclipse als IDE. Dort ermöglichen sie dem Anwender je nach Aufgabenstellung ver-

schiedene ‚Sichtweisen' auf denselben Programmcode.

Die Inhalte einer Perspektive werden wiederum über so genannte Views und Editoren in der Gestalt von ‚Karteikärtchen' strukturiert. Während Views meist nur für die Darstellung von Daten gedacht sind, dienen Editoren der aktiven Änderung von Inhalten durch den Anwender.

Wie die Views und Editoren innerhalb der Perspektive angeordnet und skaliert sind, ist zwar initial durch eine Defaulteinstellung festgelegt, jedoch sehr frei durch den Anwender veränderbar. So können Views und Editoren z.B.:

- minimiert, maximiert oder geschlossen werden.
- in einem eigenen Anwendungsfenster ‚auslagert' werden.
- skaliert werden.
- in ihrer Anordnung verändert werden.
- aus anderen Perspektiven ‚hinzugeschaltet' werden.

Jede Eclipse Anwendung kann unterschiedliche viele Perspektiven und darin wiederum unterschiedliche Views und Editoren beinhalten. Eine vordefinierte Kombination in Form von Default-Perspektiven ist dabei genauso möglich, wie eine anwendergetriebene, freie Zusammenstellung von Perspektiven. Eclipse RCP bietet damit enorme Freiräume, was die Konfiguration und Individualisierung einzelner Perspektiven anbelangt.

3.0 Usability Engineering mit Eclipse RCP anhand eines Praxisbeispiels

Im Rahmen des Entwicklungsprozesses industrieller Produkte spielen verschiedenste Arbeitsschritte eine Rolle: Von der Planung über Entwurf, Umsetzung und Verwaltung müssen verschiedenste Aufgaben in unterschiedlichen Komplexitätsgraden durchgeführt werden. Um die unterschiedlichen Bearbeitungskontexte und Anwendergruppen zu unterstützen, existieren häufig eine Reihe eigenständiger Systeme. Die Praxis zeigt, dass die Aufgabenschritte oftmals nicht isoliert betrachtet werden können, sondern fließend ineinander übergehen oder voneinander abhängen. Die Folge sind häufige Systemwechsel: Beispielsweise wird eine Aufgabe im System A begonnen, muss dann im System C referenziert und über System D einem Kollegen zugewiesen werden. Der Anwender wird bei dieser Konstellation nicht nur zahlreiche Male aus dem Bearbeitungskontext herausgerissen, er muss sich auch mit verschiedenen Anwendungsoberflächen und Bedienphilosophien auseinandersetzen. Oftmals funktionieren auch Schnittstellen zur Datenübertragung zwischen den Programmen nur bedingt, so dass z.B. Produkt-IDs aus einem System auf Papier notiert werden müssen, um in einem anderen eingefügt zu werden.

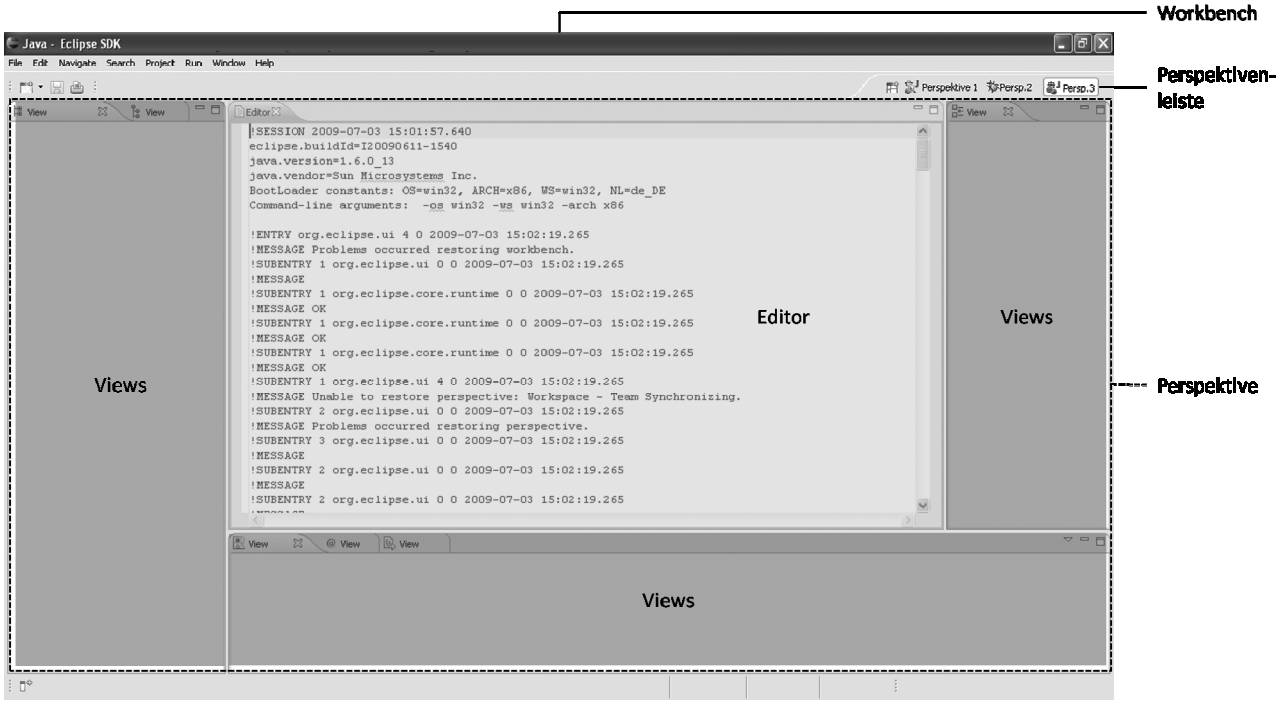

Abb. 1: Beispieldarstellung einer Eclipse RCP-Oberfläche

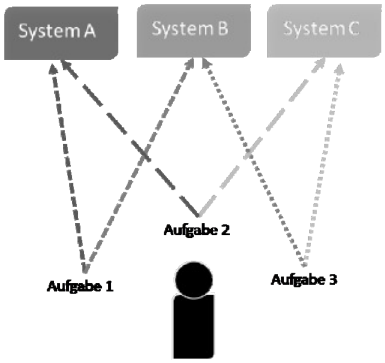

Abb. 2: Projektausgangslage - der Anwender benötigt verschiedene Systeme für die Erfüllung einzelner Aufgaben und muss zwischen Systemen wechseln.

Um die Problematik der häufigen Systemwechsel und die damit verbundenen Schwierigkeiten im konkreten Projektbeispiel zu überwinden, wurde hier beschlossen, die bislang eigenständigen Systeme in einem zentralen Client auf Basis von Eclipse RCP zu bündeln, der über Plugins Zugriff auf die verschiedenen Applikationen bietet. Ziel und Motivation für die Zusammenführung der Systeme waren:

- Einmaliges Anmelden durch den Anwender für alle Systeme (Single Sign-on)
- Vereinfachen von Systemübergreifenden Prozessen, vor allem der Vermeidung von Systemwechseln für den Anwender
- Umsetzen einer einheitlichen Bedienphilosophie und eines gemeinsamen Erscheinungsbildes

Als Vision für die zukünftige Verwendung des Systems dient eine – auf die Bedürfnisse und Aufgaben des jeweiligen Anwenders abgestimmte – Anwendungsoberfläche, die über Perspektiven genau die Informationen und Werkzeuge bereit stellt, die der Anwender für die Aufgabenbearbeitung benötigt ohne das Systemgrenzen überwunden werden müssen.

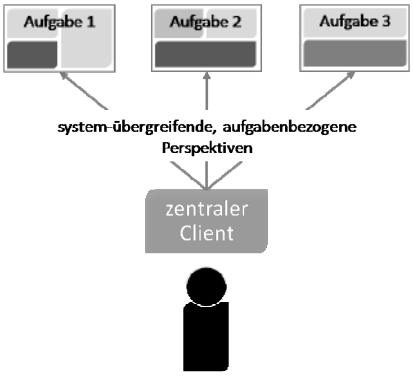

Abb. 3: Projektvision - alle notwendigen Informationen werden aufgabenbezogen in einer Perspektive gebündelt und ohne Systemwechsel bereitgestellt.

3.1 Anwendung des Perspektivenkonzepts

Bedingt durch die Größe der potentiellen Anwendergruppe, sowie deren unterschiedliche Aufgaben, ergab sich in unserem Projektbeispiel ein sehr heterogenes Anforderungsbild an die Gestaltung der Benutzungsoberfläche:

- Aufgaben, deren Bearbeitung sich über mehrere Systeme erstrecken, sollten einfacher – vor allem ohne Systemwechsel – gestaltet werden.
- Anwender, die vorwiegend isoliert in einem System arbeiten, sollten durch die anderen integrierten Systeme nicht abgelenkt werden und ein möglichst ‚unverändertes' Erscheinungsbild ihres Systems vermittelt bekommen.
- Gleichzeitig sollte das Konzept so flexibel sein, dass die Anwender sich für neue, bislang unbekannte Aufgaben Oberfläche und Inhalt individuell zusammenstellen können.

Zur Erfüllung dieser Anforderungen wurde sich das flexible Perspektivenkonzept zu Nutze gemacht. Hierzu wurden drei verschiedene Arten von Perspektiven definiert:

- Anwendungsperspektiven
- Aufgabenperspektiven
- Eigene Perspektiven

3.1.1 Anwendungsperspektiven

Die Anwendungsperspektiven sind vordefinierte Perspektiven, die den Namen des ursprünglichen Systems tragen und dem Anwender eine isolierte Sicht auf das System bieten, ohne Informationen aus anderen Systemen bereit zustellen. Layout und Gestaltung sind – unter Berücksichtigung der allgemeinen Vorgaben – möglichst nah an der des Altsystems gehalten, um dem Anwender Erscheinungsbild und Arbeitsweise wie gewohnt bereit zu stellen.

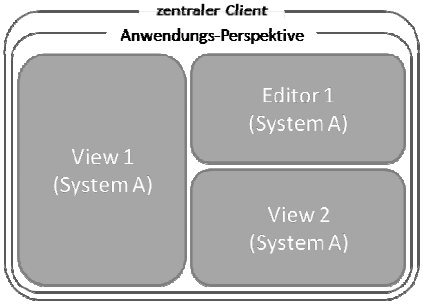

Abb. 4: Anwendungsperspektive – setzen sich allein aus Views und Editoren einer Anwendung zusammen.

3.1.2 Aufgabenperspektiven

Die Aufgabenperspektiven sind vordefinierte, gegebenenfalls systemübergreifende Perspektiven. Sie sind mit dem Namen der jeweiligen Aufgabe bezeichnet und fassen genau die Informationen aus denjenigen Systemen zusammen, die bei der Bearbeitung der Aufgabe eine Rolle spielen. Oftmals können Aufgabenperspektiven auch durch einen instruktionellen Teil ergänzt sein, der den Anwender Schritt für Schritt durch die Aufgabenbearbeitung führt und sukzessive die relevanten Systemteile einblendet.

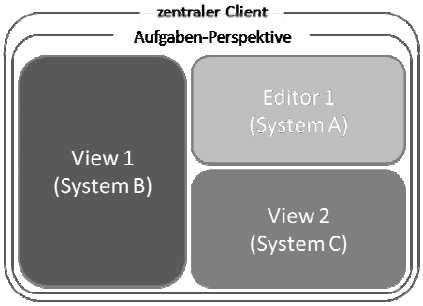

Abb. 5: Aufgabenperspektiven setzen sich aus Teilen der verschiedenen Systeme zusammen, die für die Bearbeitung der Aufgabe erforderlich sind.

3.1.3 Eigene Perspektiven

Zur Erfüllung von individuellen Aufgaben bzw. Aufgaben, die bislang noch nicht genau genug definiert werden konnten, um eine vordefinierte Perspektive für sie erstellen zu können, wurde eigens die Möglichkeit eingeführt, auch ‚Eigene Perspektiven' anlegen zu können. Eclipse RCP bietet dem Nutzer die Möglichkeit, bestehende Perspekt-iven durch Veränderungen des Layouts sowie durch Hinzufügen oder Entfernen von Views so anzupassen, dass sie den individuellen Bedürfnissen entsprechen. Diese selbst zusammengestellten Perspektiven können abgespeichert und so jederzeit erneut aufgerufen werden.

3.2 Herausforderungen bei der Arbeit mit Eclipse RCP

Die hohe Flexibilität von Eclipse RCP in Bezug auf die Zusammenstellung der Views und Editoren im Rahmen von Perspektiven wurde zuvor bereits als ein entscheidender Vorteil von Eclipse RCP genannt. Gleichzeitig liegt jedoch genau in dieser Flexibilität eine der größten Herausforderungen für Usability Professionals bei der Arbeit mit Eclipse RCP begründet. Im konkreten Beispielprojekt haben sich im Laufe der Konzeption z.B. folgende Themen herauskristallisiert:

- Gestaltung von Views und Edioren ohne fest definierbaren Verwendungsrahmen
- Verwendung von Editoren bei Anwendungen mit sehr unterschiedlichen Bearbeitungsinhalten
- Erweiterbarkeit der Eclipse Plattform als Gegensatz zur notwendigen Anpassung auf den individuelle Nutzungskontext einer Anwendung

3.2.1 Gestaltung der Views

Views können in Eclipse RCP Anwendungen sehr frei in verschiedenen Bearbeitungskontexten (Perspektiven) eingebunden werden. Sei dies nun innerhalb von eigens vordefinierten Perspektiven oder auch im Rahmen der Definition einer ‚Eigenen Perspektive' durch den Anwender. Eine der größten Herausforderungen in unserem Projekt lag daher in der Gestaltung der Views, da bislang keinerlei Erfahrungen darüber vorlagen, inwieweit welche Views aus den verschiedenen Systemen miteinander kombiniert werden würden. Für die Konzeption der Views und ihres Inhalts bedeutet dies einen nicht definierten Anwendungskontext: Weder die Position auf dem Bildschirm noch die umgebenden Inhalte können gänzlich definiert werden. Bei der Gestaltung der Views sollte daher auf folgende Aspekte besonderen Wert gelegt werden:

- Modulare Gestaltung der Views: da diese in Eclipse nicht auf den Kontext ihrer Stamm-Perspektive festgelegt sind und auch in eine andere integriert werden können aber dabei dennoch selbsterklärend sein müssen
- Eindeutige Kennung der Views beispielsweise durch einen aussagekräftigen Namen, der die View gegebenenfalls auch noch in den übergeordneten Anwendungskontext einordnet, sowie ein selbsterklärendes, individuelles Icon.
- Zusätzliche Instruktionen innerhalb der View
- Die Kombinationslogik für Views sollte unter Umständen eingeschränkt werden, das heißt, nicht jede View sollte in jede Perspektive integriert werden können, falls sich daraus Fehlerquellen ergeben könnten. Darüber hinaus kann es Komplexe aus zwei oder mehr Views geben, die nur im Verbund Sinn bereiten (z.B. Views mit Navigationscharakter und zugehörige Views zur Inhaltsanzeige) und für die daher eine isolierte Verwendung unterbunden werden sollte.

3.2.2 Verwendung von Editoren

Die Editoren in Eclipse RCP bringen eine Reihe komfortabler Funktionen mit sich: Beispielsweise eine automatische Speicherabfrage beim Schließen der Perspektive oder eine ‚dirty'-Kennzeichnung. Dennoch stellten sich die Editoren in unserem Projekt als ein entscheidender Stolperstein heraus. Bedingt durch die Wurzeln von Eclipse als IDE weisen die Editoren eine besondere Eigenschaft auf: sie bleiben vom Perspektivenwechsel unbeeinflusst, das heißt beim Aufruf einer anderen Perspektive wechselt der Editor der zuvor geöffneten Perspektive mit.

Im Kontext einer IDE ist dieses Verhalten ideal, denn:

- bearbeitet wird immer derselbe Inhalt (der Quellcode).
- die Perspektiven dienen lediglich als unterschiedliche Sichtweisen auf denselben Code.

Ein ‚Mitgehen' des Editors in den unterschiedlichen Perspektiven unterstützt den Software-Entwickler damit optimal, da er in jeder Perspektive in der Lage ist den Quellcode zu editieren.

In einem anderen Anwendungskontext, wie dem unseres Projektbeispiels, birgt dieses Verhalten jedoch Usability Probleme: Verfügt eine Eclipse RCP Anwendung zwar über diverse Perspektiven, jedoch über sehr unterschiedliche Dateninhalte, die in den Editoren bearbeitet werden sollen, leidet die Selbstbeschreibungsfähigkeit der Anwendung unter dem ‚Mitgehen' des Editors. Ohne dass der Anwender dies explizit ausgeführt hätte, wechseln Editoren aus einer Perspektive in die andere mit. Die Editoren stellen sich hier gegebenenfalls in einem völlig anderen Anwendungskontext dar und bergen die Gefahr von Fehlinterpretationen durch den Anwender.

Da die Dateninhalte der einzelnen Editoren in unserem Beispielprojekt von sehr unterschiedlicher Art sein können, wird aktuell über einen Eingriff in das Eclipse RCP Standardverhalten nachgedacht, um das Wechselverhalten abschalten oder zumindest anwendergesteuert beeinflussen zu können.

3.2.3 Erweiterbarkeit vs. individueller Nutzungskontext

Die Eclipse Plattform erhebt an sich den Anspruch einer universellen Plattform: „…an open, extensible IDE for anything, but nothing in particular." (Eclipsepedia 2009). Um diese Universalität und vor allem die Erweiterbarkeit einer Eclipse RCP Anwendung mit den immer zahlreicher werdenden Plugins gewährleisten zu können, stellt die Eclipse Foundation UI Guidelines für die Gestaltung von neuen Eclipse RCP Anwendungen zur Verfügung (www.eclipse.org/articles/Article-UI-Guidelines/Contents.html). Die dort festgelegten Richtlinien sollen auch gewährleisten, dass Erscheinungsbild und Bedienphilosophie einer Eclipse RCP Anwendung auch bei späterer Integration von ‚Fremd'-Plugins aus der Eclipse-Community konsistent und einheitlich bleiben.

Die Eclipse UI Guidelines basieren bzw. referenzieren zum Großteil auf die Windows Guidelines, was es in der Regel nicht schwer macht, sich an sie zu halten. Dennoch können sich aus dem jeweiligen individuellen Nutzungskontext einer zu entwickelnden Anwendung Anforderungen ergeben, die eine Abweichung kleineren oder größeren Ausmaßes von den Eclipse UI Guidelines erfordert. Eine individuelle Anpassung von Eclipse RCP Anwendungen ist in solchen Fällen – verbunden mit den jeweiligen entwicklungstechnischen Aufwänden – meist möglich. Im Gegenzug muss man jedoch mit dem Nachteil rechnen, dass eine spätere Integration ‚klassischer' Eclipse-Plugins:

- entweder einer erneuten Anpassung in den veränderten Aspekten bedarf
- oder einen ‚Bruch' hinsichtlich Erscheinungsbild oder Bedienphilosophie aufweist.

Aus Sicht des Usability Professionals steht in diesem Punkt daher immer die Gefahr einer Entscheidung für die Eclipse-Konventionen und gegen eine individuelle Anpassung an den Nutzungskontext der Anwendung im Raum.

4.0 Fazit und Ausblick

Durch das Perspektivenkonzept ist Eclipse RCP prädestiniert für die Umsetzung von Anwendungen mit:

- mehreren parallel zu bearbeitenden Aktivitäten.
- unterschiedlichen Sichtweisen auf dieselben Inhalte

Insgesamt sind die Möglichkeiten in Eclipse aus Sicht der Autoren nach wie vor stark auf den Kontext einer IDE fokussiert. Eine projektspezifische Anpassung der Eclipse-Oberfläche an den Nutzungskontext und die Bedürfnisse der jeweiligen Anwender ist mit den Standard-Mitteln oftmals nur bedingt möglich. Nutzerorientierte Alternativgestaltungen können dagegen bedingt durch die ungleich höheren Entwicklungsaufwände nicht immer umgesetzt werden, was letztendlich dazu führt, dass oftmals auf Standardlösungen zurückgegriffen wird.

Ein möglicher Ansatz diese Problematik aufzulösen, liegt in der Eigenschaft von Eclipse RCP als Open-Source-Framework. Vergleichbar mit den zahlreichen Plugins, die für Eclipse in der Zwischenzeit zur Verfügung stehen, könnten auch entsprechende UI-Komponenten, die ein andersartiges Navigationskonzept durch den teils schwierigen Perspektiven-Dschungel ermöglichen, öffentlich zugänglich gemacht werden und somit das Erscheinbild und die Vielfalt in Hinblick auf das Bedienkonzept von Eclipse-Anwendungen wesentlich erweitert werden.

Literaturverzeichnis

des Rivieres, J. ; Beaton, W. (2006): Eclipse Platform Technical Overview. http://eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html [12.07.2009]

Eclipsepedia (2009): User Interface Guidelines. http://wiki.eclipse.org/index.php/User_Interface_Guidelines [20.05.2009]

Ebert, R. (2009): Eclipse RCP Buch. http://www.ralfebert.de/rcpbuch/ [12.07.2009]

Usability Evaluation