HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Scala Search IDE-Integration

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2016

Autoren:	Florian Merz, Royce Manavalan
Betreuer:	Mirko Stocker
Experte:	Leo Büttiker
Gegenleser:	Prof. Dr. Farhad Mehta

1 Abstract

Die meisten Softwareprojekte bauen auf Frameworks, Bibliotheken oder bestehendem Code auf. Der Entwickler kennt die gängigsten Funktionen der oben genannten Code-Ressourcen. Das Auffinden von Funktionen ist keine leichte Aufgabe, denn es erfordert ein vertieftes Wissen über die Ressourcen oder aber man erleichtert sich die Suche mit einem Tool. Während die Standard-Tools von Eclipse nach Keywords suchen, gibt es in Scala eine Möglichkeit, in vordefinierten Bibliotheken die Funktion nach Typ-Signatur und Keywords über eine Webapplikation zu finden. Diese Webapplikation nennt sich Scaps – www.scala-search.org. Die Arbeit besteht darin, die Suchmaschine in Eclipse mittels Plugin zu integrieren, so können auch lokale Projekte und Bibliotheken von Drittanbieter zum Index hinzugefügt werden. Jedoch werden nur Bibliotheken mit angehängtem Sourcecode beachtet.

Das Plugin verwendet zwei Indexe, so wird sichergestellt, dass während des Indexiervorgangs ein Index weiterhin für die Suche verwendbar ist, während der andere Index aufgebaut wird. Im Anschluss des Indexiervorgangs werden die zwei Indexe ausgetauscht.

Die Integration von Scaps in die Scala IDE beinhaltet eine Möglichkeit, die Projektressourcen für den Suchindex auszuwählen, einen Suchdialog und einen Quickfix, welcher einen Queryvorschlag für die gesuchte Funktion macht. Mit dieser Integration der Suchmaschine wird ein besserer Workflow des Entwicklers erreicht.

2 Management Summary

2.1 Ausgangslage

Im Alltag eines Programmierers kommt es öfters vor, dass er während der Entwicklung eine Funktion sucht, die ihm eine Arbeit abnimmt. Eine solche Suche kann sich als schwierig und zeitintensiv erweisen, da verschiedene Klassen und Bibliotheken durchsucht werden müssen, um eine geeignete Funktion zu finden. Abhilfe schafft die Webapplikation Scala-Search «Scaps». Diese ermöglicht Suchanfragen sowohl mit Typ-Signaturen als auch mit Keywords. Scaps verbessert die bisherigen Suchmöglichkeiten wesentlich, da man spezifischere Suchanfragen stellen kann. Eine Einschränkung ist, dass zurzeit nur vordefinierte Bibliotheken durchsuchbar sind und Scaps nur als Webapplikation zur Verfügung steht. Die Aufgabe besteht nun darin, Scaps in die Scala IDE zu integrieren, damit eigene Projekte indexiert und durchsucht werden können, um so den Entwicklungsprozess zu verbessern.

2.2 Vorgehen, Technologien

Um den Einstieg in die Eclipse-Plugin-Entwicklung zu erleichtern, wurde der Prototype in Java entwickelt. Nebenbei haben wir uns mit Scala und Scaps auseinandergesetzt. In der nächsten Phase wurde in die Scala-IDE gewechselt, wobei das Gelernte in Scala umgesetzt wurde. Schritt für Schritt wurde Scaps in das Plugin integriert. Der erste Schritt war, dass die eigenen Projekte indexiert werden können, danach wurde Scaps ins Userinterface integriert. Mit den dargestellten Suchresultaten wurde ein erster Meilenstein erreicht. Basierend auf dem Grundgerüst wurde ein Dialog erstellt, der dem Entwickler die Möglichkeit bietet, die zu indexierenden Dateien und Bibliotheken auszuwählen. Um die Benutzerfreundlichkeit zu verbessern benutzt unser Plugin zwei Indexe, einen aktiven für die Suche und einen passiven für die Indexierung, diese werden nach Abschluss des Indexiervorgangs ausgetauscht. Damit wird sichergestellt, dass zu jedem Zeitpunkt eine Suche ausgeführt werden kann.

2.3 Ergebnisse und Ausblick

Das Ergebnis ist ein funktionsfähiges und nützliches Plugin für die Scala-IDE. Die einfache Benutzeroberfläche ermöglicht dem Entwickler einen besseren Workflow. Das Plugin kann als Grundlage für weitere Features gebraucht werden:

- Eine Eingabehilfe, die es dem Entwickler ermöglicht, eine Query aus den Variablen, die im Kontext vorhanden sind, zu bauen.
- Kontinuierliche Indexierung: Der Indexer wird automatisch neu gebaut, wenn sich genügend Ressourcen geändert haben.
- Mehrere Indexe: Der Entwickler kann mehrere Indexe anlegen.
- Java: Scaps wird um die Indexierung von Java Sourcen erweitert.

3 Inhaltsverzeichnis

4 Aufgabenstellung

Aufgabenstellung Bachelorarbeit Royce Manavalan und Florian Merz „Scala Search IDE-Integration“

1. Betreuer und Experte

Diese Bachelorarbeit wird für das Institut für Software (IFS) der HSR durchgeführt.

Betreuer:

- Mirko Stocker, HSR, IFS, m1stocke@hsr.ch (verantwortlicher Dozent für Beurteilung)
- Lukas Wegmann, HSR, IFS l1wegman@hsr.ch

Experte:

- [zu bestimmen]

2. Studierender

Diese Arbeit wird als Bachelorarbeit an der Abteilung Informatik durchgeführt von

- Royce Manavalan, rmanaval@hsr.ch
- Florian Merz, fmerz@hsr.ch

3. Einführung

Scaps - http://scala-search.org - ist eine Suchmaschine die Definitionen in Scala Bibliotheken anhand von Keywords und Typ Signaturen findet. Zur Zeit ist Scaps als Webservice verfügbar und erlaubt das Durchsuchen von vordefinierten Bibliotheken. Scaps kann auch lokal installiert werden, um eigene Projekte zu indexieren und zu durchsuchen.

4. Ziele der Arbeit

Das Ziel dieser Arbeit ist, Scaps in die Scala IDE zu integrieren. Zum einen sollte dadurch das Benutzen von Scaps in eigenen Projekten vereinfacht werden. Und anderseits kann, durch das einbeziehen von Kontextinformationen aus dem Editor, die Genauigkeit der Suchmaschine weiter verbessert werden.

5. Aufgabenstellung

Folgende Arbeitsschritte sind Teil des Projekts:

1. Setup der Entwicklungsumgebung mit Eclipse und der Scala IDE
2. Integration der Suchmaschine in die IDE: Die Projektinformationen aus der IDE sollen genutzt werden um den Index aufzubauen.
3. Entwickeln des Such UIs: Hier sind innovative Ideen gefragt, wie die Suche optimal in den Workflow eines Entwicklers eingebetet werden kann. Die Suche sollte, ähnlich wie Code Completion, direkt aus dem Editor gestartet werden können.

Je nach Interesse und Projektfortschritt können auch weitere Aspekte betrachtet werden:

1. Iteratives Indexieren: Damit der Index immer den aktuellen Stand des Projekts abbildet, muss dieser laufend Aktualisiert werden. Zur Zeit können nur ganze Projekte indexiert werden, was ein zeitnahes indexieren verunmöglicht.
2. Byte Code Extraktor: Scaps parst Scala Source Files um den Index aufzubauen. Falls diese nicht vorhanden sind, können Bibliotheken nicht durchsucht werden. Als Fallback könnte ein Byte Code Parser umgesetzt werden, der Definitionen aus Java Class Files extrahiert.

6. Zur Durchführung

Es finden wöchentliche Besprechungen mit den Betreuern statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen.

Alle Besprechungen, ausser der Kick-off Besprechung, sind von den Studierenden mit einer Traktandenliste vorzubereiten und zu leiten. Als erstes Traktandum soll immer der Stand des Projektes präsentiert werden (Was wurde gemacht? Was wurde erreicht? Wie viel Zeit wurde in was investiert?). Die Ergebnisse der Besprechungen sind durch die Studierenden zu protokollieren und an die Betreuer anschliessend zuzustellen.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen (oder auch Zwischenversionen) gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Resultate.

7. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen (siehe https://www.hsr.ch/Allgemeine-Infos-Diplom-Bach.4418.0.html). Die zu erstellenden Dokumente bzw. Berichtsteile sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Alle Resultate sind vollständig auf CD/DVD in 3 Exemplaren abzugeben.

8. Termine

Siehe auch Terminplan auf https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html.

Montag, den 22. 2. 16	**Beginn der Bachelorarbeit**
8.6.16	Abgabe Kurzbeschreibung und A0-Poster an Betreuer zur Überprüfung Vorlagen stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung.
15.6.16, 10.00	**Fertigstellung des A0-Posters bis 10.00 Uhr und Weiterleitung als PDF-Dokument an das Studiengangsekretariat.**

17.6.16, 12.00	**Abgabe der Arbeit an den Betreuer bis 12.00 Uhr.**
17.6.16, 16.00	Präsentation der Bachelorarbeiten, 16 bis 20 Uhr
8.8. - 26.8.16	Mündliche Prüfung zur Bachelorarbeit
30.9.16, 9.00	Bachelorfeier und Ausstellung der Bachelorarbeiten

9. Beurteilung

Eine erfolgreiche Bachelorarbeit zählt 12 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von 30 Stunden budgetiert (Siehe auch Modulbeschreibung der Bachelorarbeit https://unterricht.hsr.ch/staticWeb/allModules/19419_M_BAI.html.).

Für die Beurteilung ist der verantwortliche Dozent zuständig.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/6
2. Berichte (Abstract, Mgmt Summary, technischer u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/6
3. Inhalt*)	3/6
4. Mündliche Prüfung zur Bachelorarbeit	1/6

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit präzisiert.

Im Übrigen gelten die Bestimmungen der Abteilung Informatik für Bachelorarbeiten.

Rapperswil, den 11. Februar 2016

M. Stocker

Mirko Stocker

Dozent für Informatik
Institut für Software (IFS)
Hochschule für Technik Rapperswil

5 Eigenständigkeitserklärung

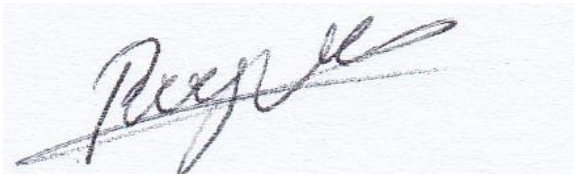
Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- dass ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum: Rapperswil, 15.06.2016

Name, Unterschrift:

Florian Merz
15.06.2016, Rapperswil



Royce Manavalan
15.06.2016, Rapperswil

6 Danksagung

Unser Dank geht an Mirko Stocker und Lukas Wegmann für die Betreuung der Bachelorarbeit. Die Freiheit, die sie uns in der Projektrichtung gaben, ermöglichte uns, eigene Ideen in das Projekt einzubauen.

7 Aufteilung

Die Suchmaschine Scaps wurde im Rahmen der Masterarbeit von Lukas Wegmann entwickelt. Scaps ist die Grundlage, auf der unsere Bachelorarbeit aufbaut. Der ganze Indexier und Suchvorgang wird durch die Scaps-Bibliothek gehandhabt.

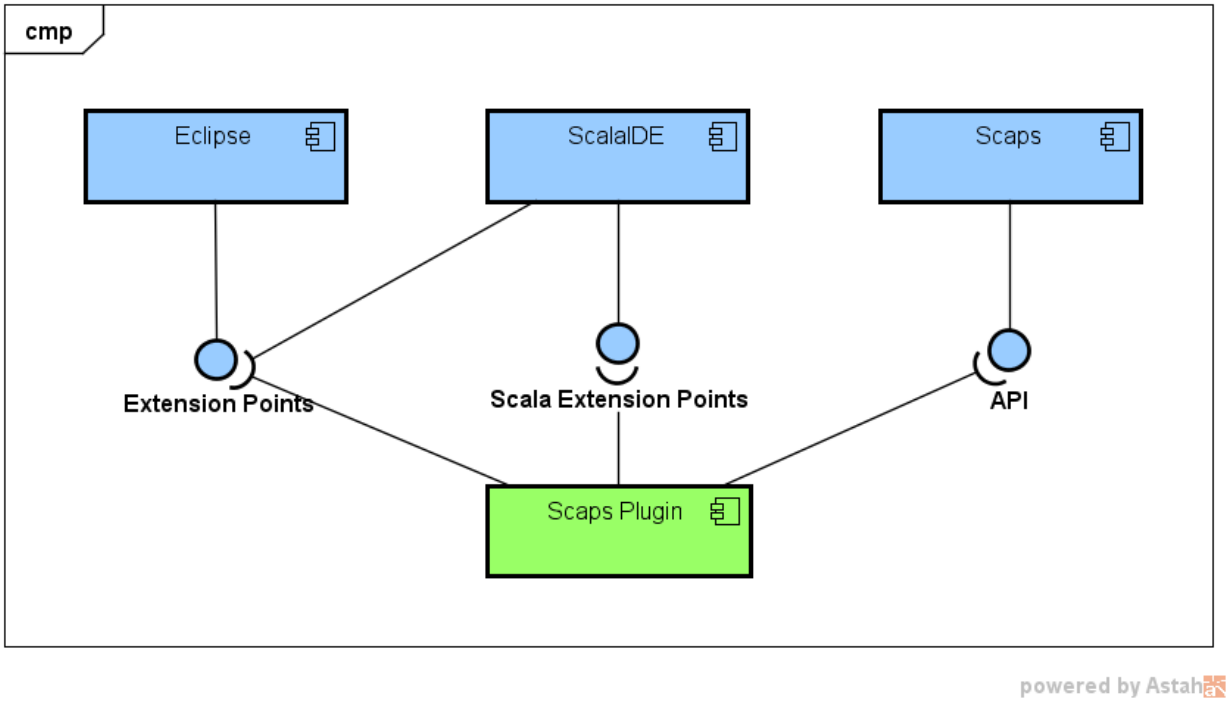


Abbildung 1: Komponentendiagramm Scaps Eclipse

Wir entwickelten das Plugin für die Scala-IDE, welches grundsätzlich als Userinterface dem Benutzer zur Verfügung steht. Das Plugin hat zwei Aufgaben, auf der einen Seite arbeitet es als Schnittstelle zur Scaps-API, welche den logischen Teil übernimmt, auf der anderen Seite haben wir das Eclipse Userinterface, mit welchem der Entwickler kommuniziert.

8 Technischer Bericht

In diesem Kapitel gehen wir auf die technischen Details der Applikation ein. Nach einer Analysephase, in der die Use Case Diagramme und nichtfunktionalen Anforderungen festgelegt wurden, wird kurz auf das externe Design eingegangen. Danach folgt die Umsetzung mit den Details zu den einzelnen Use Cases. Im Anschluss wird die Softwarearchitektur erläutert. Zum Schluss werden das verwendete Framework, die Tools und das Testing beschrieben.

Scaps ist eine Suchmaschine, die als Webapplikation unter www.scala-search.org erreichbar ist. Sie ermöglicht dem Benutzer Funktionen anhand von Keywords und Typ Signaturen zu finden.

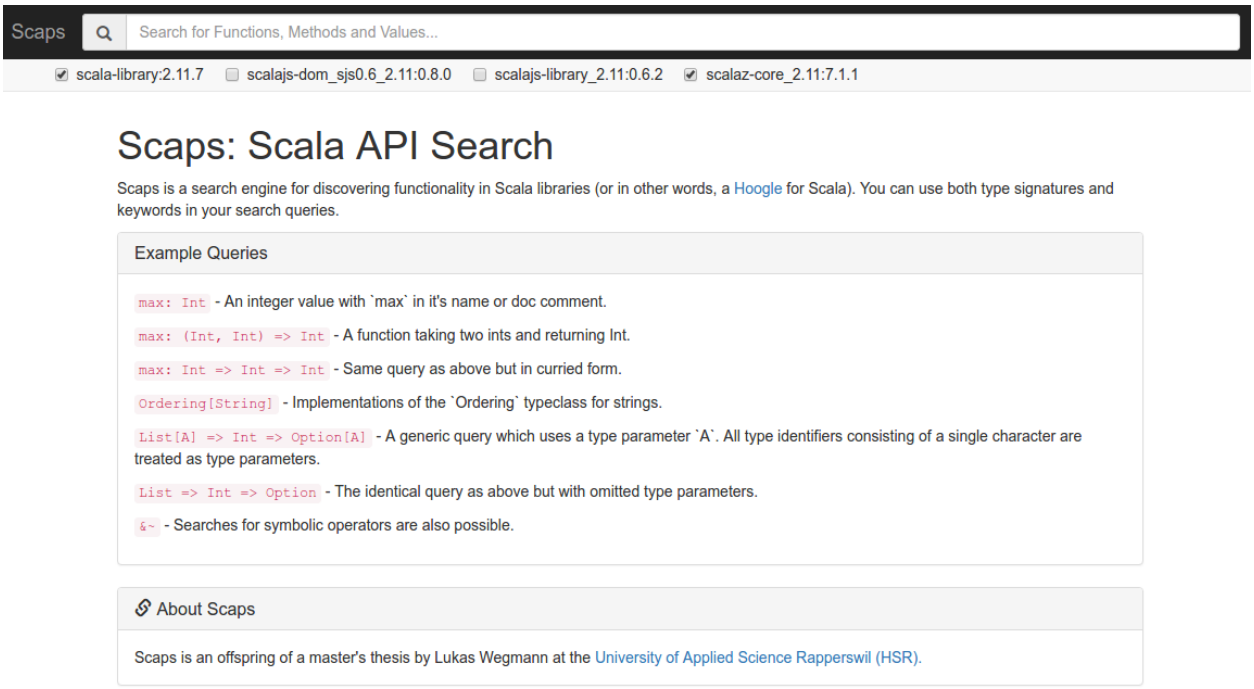


Abbildung 2: Screenshot Webapplikation Scaps

8.1 Ausgangslage

Der Entwickler hat heutzutage mehrere Möglichkeiten, nach einer Funktion zu suchen, zu den häufigsten gehören die Folgenden: Die erste Möglichkeit besteht darin, über die Eclipse Suche nach dem Funktionsnamen zu suchen, beziehungsweise über die Resourcen etwas manuell zu suchen. Die zweite Möglichkeit ist, nach einer bestehenden Lösung im Internet zu suchen, dabei muss man auf eine gute Wortwahl achten. Hier ist es schwierig, genau die richtigen Keywords zu finden, da es keine einheitlichen Namensdefinitionen gibt. Die dritte Möglichkeit ist die Webapplikation Scaps. Mit Scaps kann man nach Namen, Parametertypen und Ausgabetypen suchen. Bei dieser Möglichkeit kann man nicht viel falsch machen, da der Typ einzigartig ist und zu bestimmt anwendbaren Ergebnissen führt. Diese Webapplikation ist eine grosse Hilfe für den Entwickler, jedoch ist die Suchmöglichkeit auf vordefinierte Bibliotheken limitiert und nicht in der Entwicklungsumgebung vorhanden.

8.2 Problembeschreibung

Scaps soll nicht nur als Webapplikation mit begrenzter Anzahl Bibliotheken als Suchressourcen zur Verfügung stehen, sondern in die Scala-IDE integriert werden. Damit kann der Entwickler auch die eigenen Projekte indexieren. Durch die Integration von Scaps in die IDE, stehen dem Entwickler grössere Trefferquoten zur Verfügung, da nun auch andere Bibliotheken durchsucht werden können. Eine benutzerfreundliche Suchmaske soll ihm dabei helfen, eine Funktion zu finden. Eine Produktionssteigerung wurde erreicht, durch welche dem Entwickler eine Suchquery basierend auf der Codezeile innerhalb des Codeeditors in einem bekannten Quickfix-Menü vorgeschlagen wird.

8.3 Anforderungsspezifikation

8.3.1 Use Case Diagramm

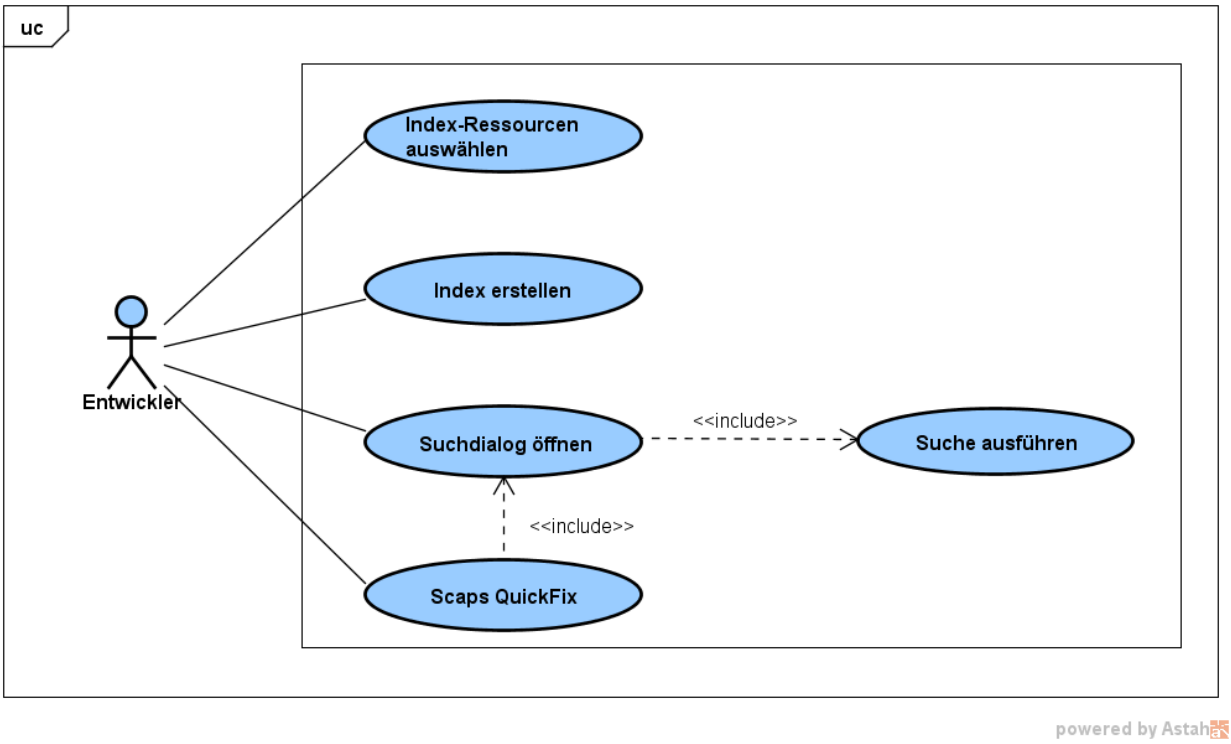


Abbildung 3: Use Case Diagramm

Das oben aufgeführte Diagramm zeigt die Use Cases, welche implementiert wurden. Sie werden nachfolgend in Fully-Briefed Use Case beschrieben.

8.3.2 Index-Ressourcen auswählen

Primary Actor	Entwickler
Stakeholders and Interests	Entwickler: wählt die Ressourcen, die zu indexieren sind, aus.
Preconditions	Es existieren Ressourcen in der Workspace.
Successs Guarantee / Postcondition	Indexer kennt die zu indexierenden Ressourcen.
Main Success Scenario	1. Benutzer ruft den Dialog auf. 2. System zeigt den Dialog. 3. Benutzer selektioniert Ressourcen. 4. Benutzer drückt auf «Finish» 5. System speichert die ausgewählten Ressourcen für den Indexer
Extensions	3a. Benutzer möchte den Index nicht sofort bauen.
Frequency of Occurence	
Open Issues	

8.3.3 Index erstellen

Primary Actor	Entwickler
Stakeholders and Interests	Entwickler: möchte, dass die ausgewählten Ressourcen in Scaps indexiert werden.
Preconditions	UC Index-Ressourcen auswählen
Successs Guarantee / Postcondition	Ressourcen sind mit Scaps durchsuchbar.
Main Success Scenario	1. Benutzer ruft «Run Indexer» auf 2. System startet die Indexierung als Background-Prozess
Extensions	3. Es existiert kein WorkingSet -> UC Index-Ressourcen auswählen
Frequency of Occurence	
Open Issues	

8.3.4 Suchdialog öffnen

Primary Actor	Entwickler
Stakeholders and Interests	Entwickler: möchte mit einer Query eine/mehrere Funktionen finden, die seinen Anforderungen entsprechen.
Preconditions	- Erstellen des UC Index wurde durchgeführt - Der Entwickler kennt die Query-Syntax und kann damit eine korrekte Query erstellen.
Successs Guarantee / Postcondition	Der Entwickler erhält in einer View Resultate, die den Anforderungen des Entwicklers entsprechen.
Main Success Scenario	1. Entwickler ruft den Suchdialog auf 2. System füllt Suchfeld mit zuletzt verwendetem Query 3. Entwickler gibt seine Query in das Suchfeld ein 4. UC Suche ausführen 5. System zeigt die Resultate dem Entwickler in einer View an 6. Entwickler macht einen Doppelklick auf ein Suchresultat 7. System öffnet das entsprechende File mit einer Markierung der gesuchten Funktion
Extensions	5a. Die Query des Entwicklers entspricht nicht der Syntax - es wird ein Fehler angezeigt 5b. Es werden keine Suchresultate gefunden 6a. Mit CTRL+C wird die Funktion in das Clipboard kopiert
Frequency of Occurence	
Open Issues	Was passiert, wenn der Entwickler z.B einen Doppelklick auf einer der Suchresultate macht? (Copy Method to Clipboard)

8.3.5 Suche ausführen

Primary Actor	System
Stakeholders and Interests	Entwickler**:** Der Entwickler möchte Suchresultate zu seiner Query sehen.
Preconditions	UC Index erstellen
Successs Guarantee / Postcondition	Scaps findet Resultate
Main Success Scenario	1. Query wird Scaps überreicht 2. Scaps führt die Suche durch 3. Scapsresultate werden geliefert 4. Scapsresultate werden konvertiert 5. Resultate werden dem Aufrufer übermittelt
Extensions	
Frequency of Occurence	
Open Issues	Wie werden die Scapsresultate gehandelt? Verwenden wir Converters oder Adapters?

8.3.6 Scaps Quick Assistant

Primary Actor	Entwickler
Stakeholders and Interests	Entwickler: kennt die Parameter und möchte mögliche Funktionen vorgeschlagen bekommen.
Preconditions	UC Indexer erstellen wurde durchgeführt
Successs Guarantee / Postcondition	Entwickler erhält einen Queryvorschlag im Quickfix Menü.
Main Success Scenario	1. Entwickler schreibt eine Codezeile 2. Entwickler ruft Scaps Quick Assistant auf 3. System extrahiert aus dem Kontext die Parametertypen für eine Scaps Suche 4. System bildet eine Query 5. System stellt Query im Quickfix dar 6. Entwickler entscheidet sich für den Queryvorschlag 7. System speichert Query als zuletzt verwendete Query ab 8. UC Suchdialog öffnen 9. UC Suche ausführen
Extensions	3a. System kann mit dieser Zeile keine Query erstellen 5a. Es gibt keine Suchresultate zu dieser Query 6a. Entwickler ist nicht zufrieden mit Query - lässt die Suche sein
Frequency of Occurence	
Open Issues	Wie muss die Codezeile aussehen? Welche Informationen werden extrahiert? Können auch mehrere Zeilen für den Kontext zur Hilfe genommen werden?

8.4 Nichtfunktionale Anforderungen

In diesem Kapitel werden nichtfunktionale Anforderungen an das Projekt erläutert.

8.4.1 Effizienz (efficiency) - Zeitverhalten (time behaviour)

8.4.1.1 Quick Assist

Der Quick Assist wird innert 2 Sekunden mit einem Queryvorschlag dargestellt. Im Durchschnitt benötigt der Quick Assist 1 Sekunde.

8.4.1.2 Search Dialog

Der Search Dialog wird innerhalb von 3 Sekunden dargestellt. Im Durchschnitt benötigt er ca. 1 Sekunde.

8.4.2 Übertragbarkeit (portability)

8.4.2.1 Anpassbarkeit (adaptability)

Da später Scaps auch für Java zur Verfügung stehen soll, sollte die Implementierung nicht ausschliesslich für Scala eingeschränkt werden. Dabei soll das Plugin nicht für andere Sprachen zur Verfügung stehen. (zb. C, C++, PHP)

8.4.2.2 Koexistenz (co-existence)

Das Plugin sollte keine anderen Plugins beeinträchtigen.

8.4.3 Zuverlässigkeit (reliability)

Eclipse sollte während des Indexieren immer benutzt werden können.

8.4.4 Bedienbarkeit (usability)

Das Plugin ist durch das simple User Interface selbsterklärend und ist einfach verständlich.

8.5 Externes Design

Basierend auf den Möglichkeiten, die Eclipse zur Verfügung stellt, wurden die folgenden Screens erstellt. Diese Screens dienen als Entwürfe und sind in der Implementierung nicht zu 100% zu übernehmen.

8.5.1 Scaps Search

Dieser Screen zeigt an, wie man nach einer Funktion suchen kann. Die Scaps Search wird in das Standard Search eingebunden. Die nichtfunktionale Anforderung Bedienbarkeit soll mit der Einbindung in die bisherige Suche erfüllt werden.
Beim Aufruf der Suche erscheint ein Dialog, in dem man auch den Reiter der Scaps Search sieht, beim Aufruf der Scaps Search erscheint direkt der Scaps Search-Reiter. Die Scaps Search enthält ein einfaches Eingabefeld. Der Grund dafür, dass die Suchabfrage nicht mit weiteren Suchfelder erweitern werden, ist einerseits die Komplexität und andererseits die Missverständnisse, die für den Benutzer dabei entstehen können. Das Suchfeld ist wie bei www.scala-search.org.

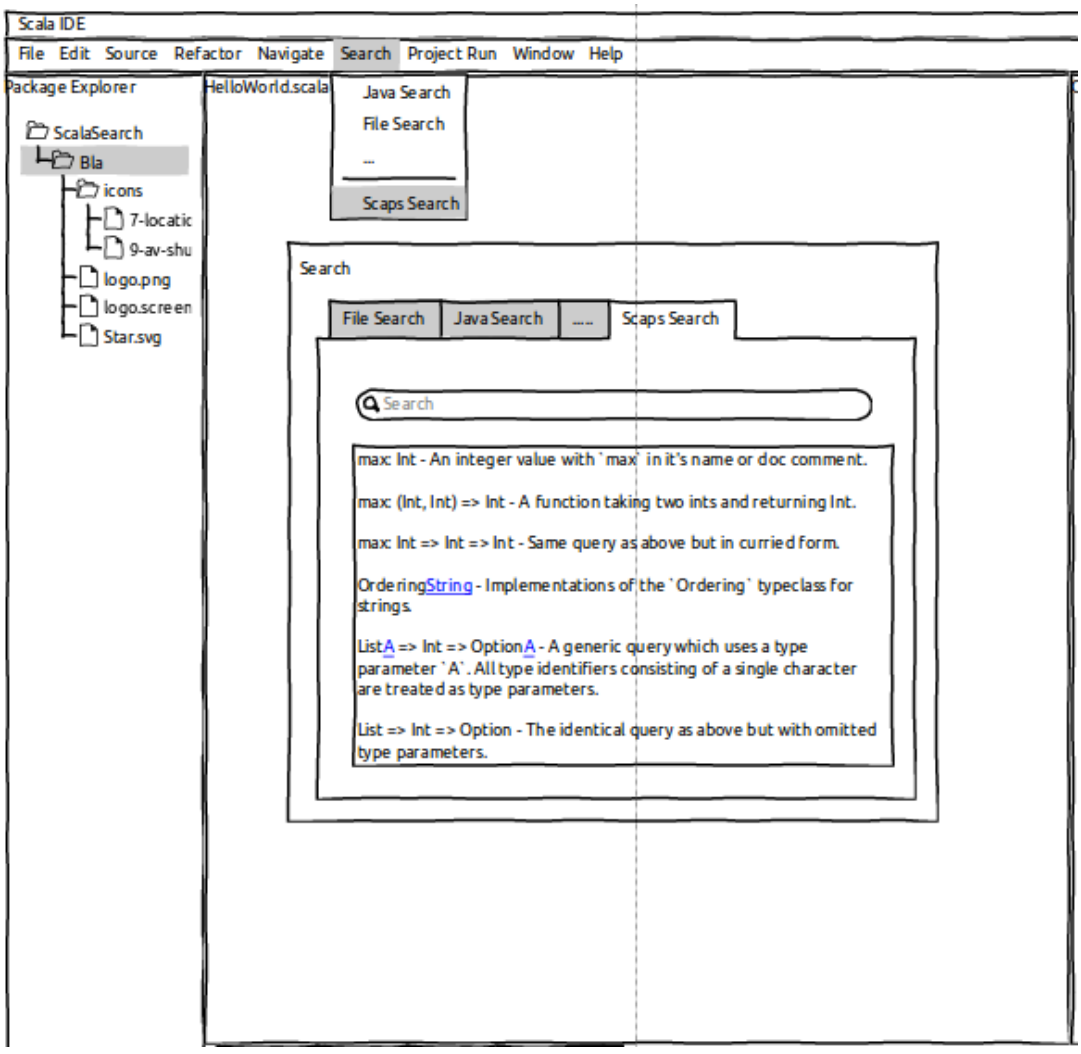


Abbildung 4: Wireframe Suchdialog

8.5.2 Scaps Assistant

Die Idee des Scaps Assistant ist es, dass der Entwickler im Editor einen Methodenaufruf mit Argumenten schreiben kann. Diese Codezeile ergibt einen Fehler, der aufgerufene Quickfix schlägt dem Entwickler verschiedene Funktionen vor.

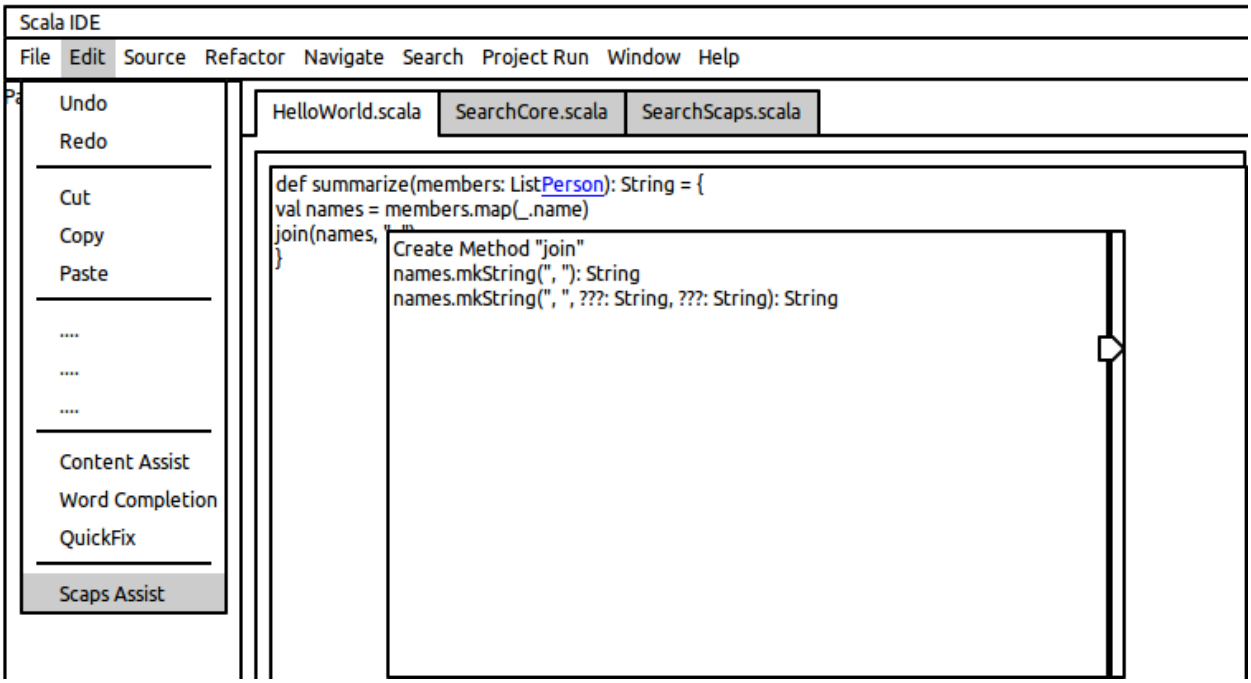


Abbildung 5: Wireframe QuickFix

Dieses Wireframe zeigt, wie der Scaps Assistant aufgerufen werden kann. Man kann den Assistenten über das Menü “Edit -> Quickfix” oder im Editor mit einem Tastenkürzel aufrufen. Beim Aufruf erscheint ein QuickFix Menü.
Während der Entwicklung kamen verschiedene Konzepte in Frage, diese werden hier erläutert:

- Aufruf von ScapsSearch mit Methodensignatur
  Beim Aufruf von Quickfix wird aus der geschriebene Funktion ein Queryvorschlag für die Scaps Suche generiert. Der Queryvorschlag wird im Hintergrund aus dem Kontext generiert.
- Ungenaue Suchresultate von ScapsSearch
  Aus dem Kontext werden die Informationen gesammelt und beim Aufruf des Quickfix erscheinen schon einige Resultate, die schneller geladen werden können, jedoch nicht zu 100% stimmen müssen. Diese Aufgabe müsste als Hintergrundtask abgewickelt werden, damit Eclipse weiterhin verwendbar ist. Damit würde auch die nichtfunktionale Anforderung mit Zuverlässigkeit erfüllt werden.

8.5.3 Scaps Search Resultview

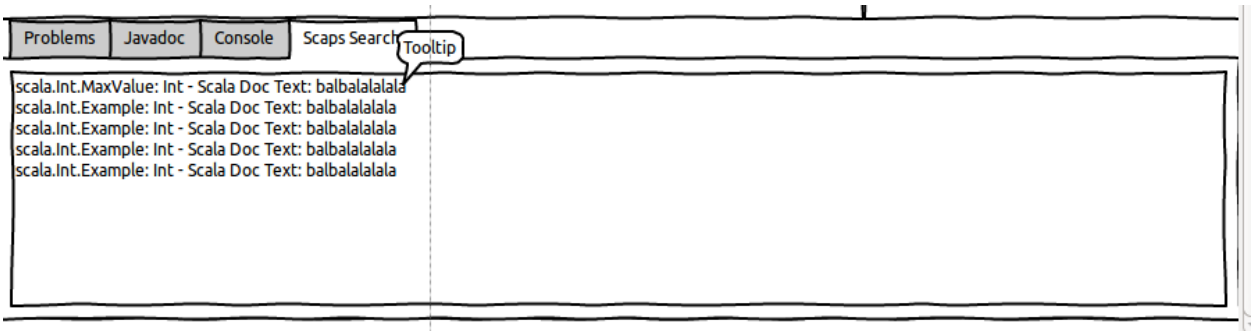


Abbildung 6: Wireframe Resultat Ansicht

Die Resultate der Suche werden in der gleichen View erscheinen wie die Resultate der Eclipse Suche. So können bestehende Kozepte von Eclipse wiederverwendet werden, dies stellt die Bedienbarkeit des Plugins sicher.
Die Suchresultate werden in der View nach Relevanz aufgelistet, bei der man die Möglichkeit hat, einen Tooltip zu bekommen.
Zu klären gibt es, was für eine Aktion bei einem Doppelklick auf eine der Suchresultate folgen soll. Zum Beispiel wäre es möglich, die Methodensignatur in das Clipboard zu kopieren, so dass der Benutzer den Code nicht abschreiben muss.

8.6 Ergebnisse

Für die Sourcecode-Versionierung haben wir Git verwendet. Dazu wurde das Projekt auf Github gehostet: https://github.com/flomerz/scala-ide-scaps

8.6.1 Use Case Realisierung

8.6.1.1 Index-Ressourcen auswählen

Die Ressourcen für den Indexer können über einen Working Set Editor ausgewählt werden. Das Konzept des Working Sets existiert in Eclipse und dient dazu, Ressourcen zu bündeln. Dies können Projekte, Libraries oder Source Ordner sein. Dieser Ansatz passt perfekt auf die Bedürfnisse von Scaps. Um einen eigenen Working Set Editor zu erstellen, muss ein neuer Working Set Typ erstellt werden. Im Fall von Scaps heisst dieser «ScapsWorkingSet». Danach kann der Typ über ein Eclipse Extension Point mit einer Working Set Editor Klasse verknüpft werden. Beim Scaps Working Set Editor werden nur Ressourcen angezeigt, welche auch indexiert werden können. Das sind alle Projekte, Source Ordner und Libraries mit Source Code Anhang. Dafür wurde ein eigener ContentProvider erstellt, der den Workspace nach diesen Ressourcen durchsucht.

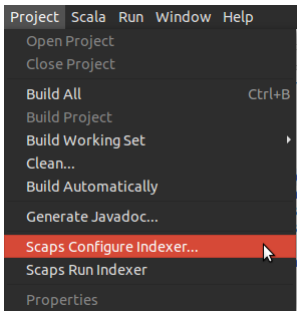


Abbildung 7: Screenshot Configure Index

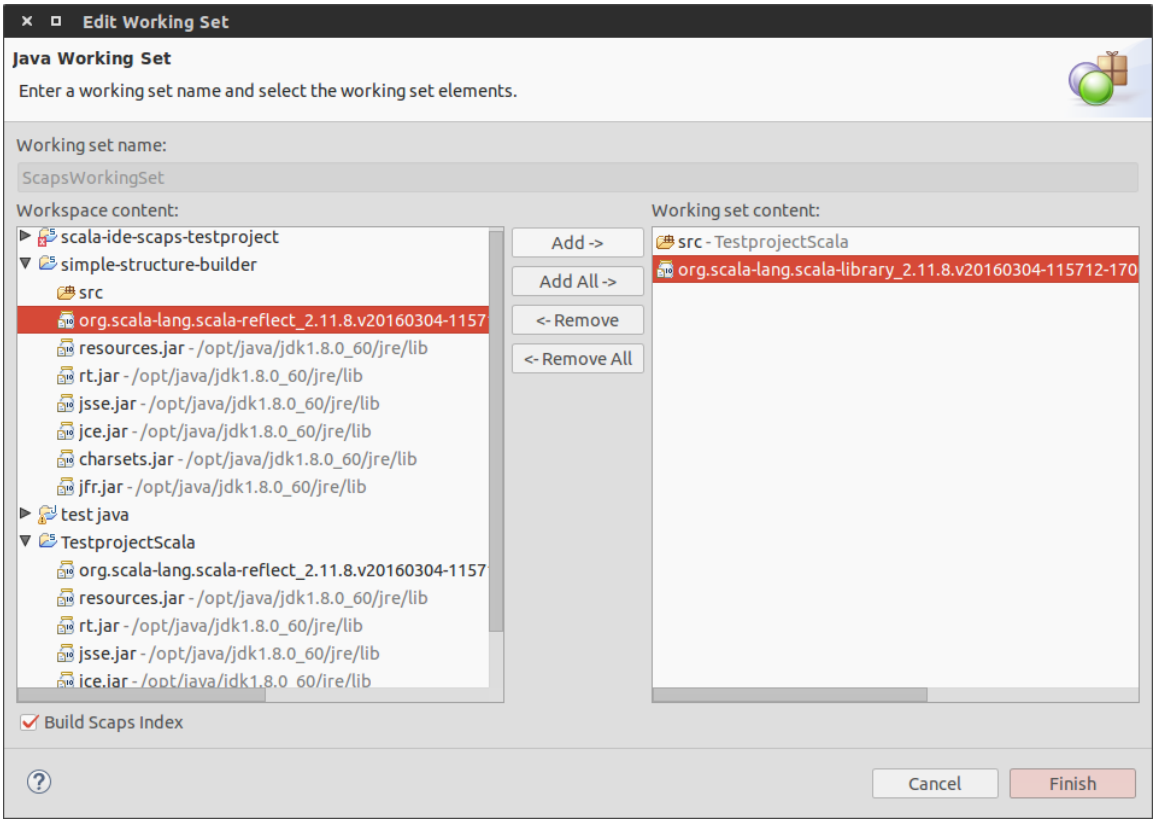


Abbildung 8: Screenshot Scaps Working Set

Probleme

Der Entwickler muss eine Möglichkeit haben, die Bibliotheken und Files auszuwählen, welche er indexieren möchte. Der erste Ansatz bestand darin, alle im Workspace vorhandenen Projekte dem Index hinzuzufügen. Dieser Lösungsansatz war jedoch nur für Testzwecke sinnvoll, da der Entwickler nicht zwingend alle Projekte indexieren will und der Indexiervorgang dadurch sehr lange dauern kann.
Innerhalb von Eclipse wurde nach bekannten Dialogen gesucht, die dem Benutzer eine bessere Auswahl von Ressourcen zur Verfügung stellt.

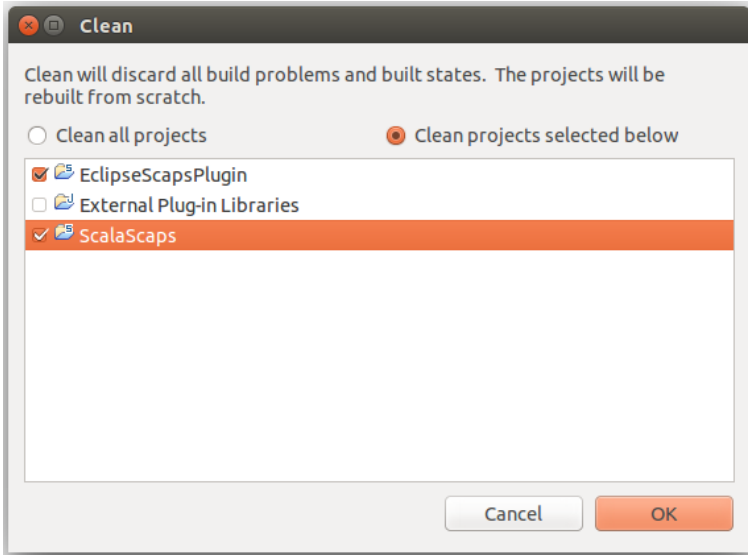


Abbildung 9: Screenshot Clean-Dialog

Der erste Treffer war der Project-Clean Dialog. Der Dialog entsprach den minimalen Anforderungen. Während der Entwicklung des Project-Clean Dialogs stiessen wir auf den New WorkingSet Dialog. Dieser Dialog stellte alle Projekte und deren Ressourcen in Tree-Form dar, was ihn zu einem geeigneteren Dialog für den Indexer machte. Also stellten wir die Entwicklung des Clean Dialogs ein und entwickelten den New WorkingSet Dialog mit einigen Erweiterungen nach.

8.6.1.2 Index erstellen

Die Ressourcen, welche ausgewählt wurden, werden indexiert. Damit das Eclipse UI während dem nicht blockiert wird, läuft der Job in einem Hintergrundprozess. Bei einem Fehler wird der Benutzer mit einem grafischen Dialog benachrichtig und falls eine Exception auftaucht, wird diese in der Eclipse ErrorLog View dargestellt. Für das Scaps Plugin gibt es zwei Scaps Indexe. Die Indexierung findet auf dem passiven Scaps Index statt, um es dem Entwickler zu ermöglichen, jeder Zeit eine Suche abzusetzen. Die Suche geht dann auf den aktiven Scaps Index. Nach Abschluss der Indexierung werden die Scaps Indexe vertauscht. Das passiert in einer synchronisierten Funktion. Diese Punkte erfüllen die nichtfunktionale Anforderung Zuverlässigkeit.

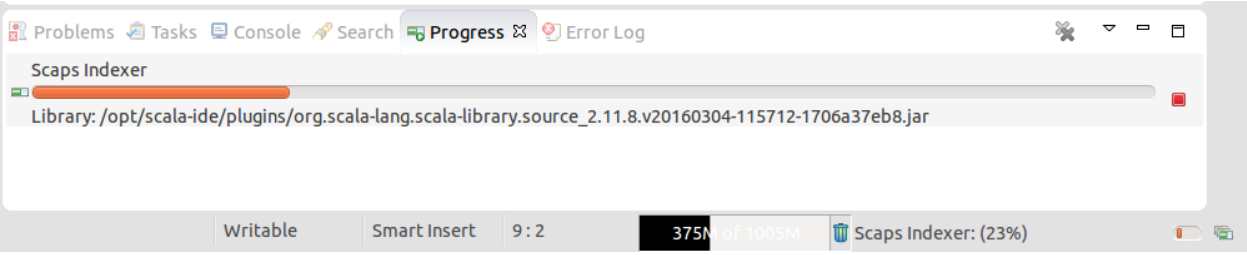


Abbildung 10: Screenshot Indexer Progress

8.6.1.3 Suchdialog öffnen

Der Suchdialog erweitert den Eclipse Suchdialog auf. Die anderen Suchseiten wie die Java Suche oder Text Suche verwenden auch den Eclipse Suchdialog als Basis. Der Aufbau ist simpel gehalten. Dem User steht ein Textfeld für die Suchanfrage zur Verfügung. Zusätzlich werden einige Beispielsuchanfragen angezeigt, welche direkt ausgeführt werden können. Damit der Entwickler seine letzte Suchanfrage bearbeiten kann, wird die Anfrage bei der Ausführung abgespeichert und bei nächsten öffnen der Suchdialogs in das Textfeld geladen.

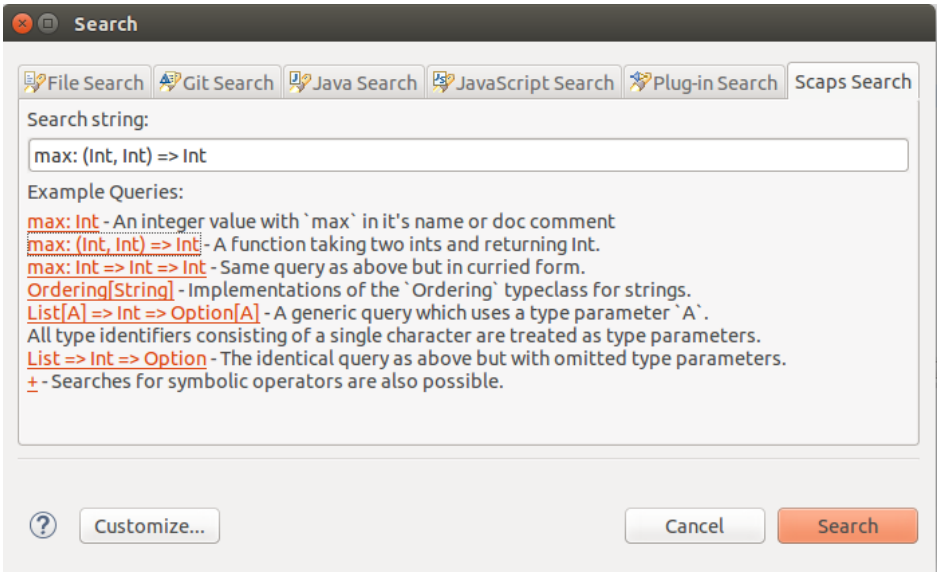


Abbildung 11: Screenshot Suchdialog

Probleme

Nach dem uns bewusst war, wie man einzelne Einträge in die Menüleiste machen konnte, wollten wir einen Eintrag für die Scaps Suche machen. Jegliche Versuche mit Handlern, Commands und Menu scheiterten. Währenddessen suchten wir im Internet nach möglichen Lösungen und siehe da, das Suchmenü wurde mit einem sogenanntem ActionSet implementiert. Das Unschöne daran war jedoch, dass das ActionSet deprecated ist. Da es inzwischen kein Update gab, mussten wir mit dem ActionSet arbeiten.

8.6.1.4 Suche ausführen

Dieser Use Case besteht aus zwei Teilen.
Zum einen wird die Suchanfrage aus dem Textfeld des Suchdialogs in eine Eclipse Suchanfrage gekapselt. Dadurch wird die Anfrage im Hintergrund ausgeführt und das Resultat der Resultatview übergeben. Das ist notwendig, um danach die Resultate in einer eigenen Resultatview anzuzeigen. Die Resultatview wird über mit dem dazugehörigen Resultattyp verknüpft. So weiss Eclipse, welche Resultatview für das Scaps Resultat angezeigt werden muss. Ein ContentProvider und ein LabelProvider bringen die Scaps Resultate in die richtige Form. Die Resultatview ist mit einer integrierten Dokumentationsanzeige ausgestattet, welche bei der Auswahl die JavaDoc der entsprechenden Funktion anzeigt. Macht der Benutzer einen Doppelklick auf eine Resultatzeile wird der Sourcecode geöffnet und die Funktion markiert. Dadurch sieht der Entwickler, ob es sich um die richtige Funktion handelt. Er kann danach die Resultatzeile kopieren und in seinem Sourcecode einfügen.
Der zweite Teil ist die Übergabe der Suchanfrage an die Scaps Suchengine und die anschliessende Auswertung sowie Fehlerbehandlung.

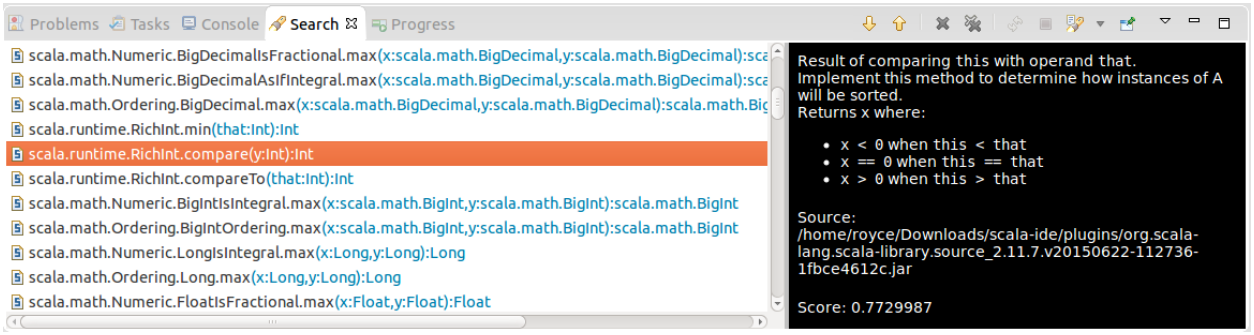


Abbildung 12: Screenshot Resultate

Probleme

Resultatview

Die Resultate der Scaps Suche sollten in einer View angezeigt werden. Am besten eignet sich dafür die ResultView die auch von Eclipse File Search verwendet wurde, so können wir Code wiederverwenden, ohne Duplicate Code zu generieren.
Die Theorie klang gut, doch bei der Umsetzung tauchten mehrere Probleme auf. Die Resultview gibt eine komplexe Struktur vor, welche eingehalten werden muss. Die Suchquery, die als String kommt, muss in ein ISearchQuery verpackt werden. Das Scapsresult muss in ISearchResult gewandelt werden, damit es auch als Resultat angezeigt wird. Wichtig ist es, dass die Resultview stets im Listmode angezeigt wird.

Ressourcen öffnen

Verschiedene Ansätze wurden verfolgt zum Öffnen der Resultate aus der Result View mit einer Markierung auf der gefundenen Funktion. Nach einigen Versuchen gelang das Öffnen und Markieren von Datei respektive der Funktion. Das grosse Problem stand aber noch bevor. Dieser Lösungsansatz war nicht zukunftsorientiert, da mit dieser Lösung keine Files von Bibliotheken geöffnet werden konnten. Die Problematik bestand darin, dass es kein fertiges Konzept gab, welches uns erlaubte, einen Pfad zu einer Bibliothek anzugeben, um diese dann im System zu verwenden. Man muss beachten, dass diese Bibliothek auch eine innere Struktur hat, in welcher das File letztendlich gefunden werden muss. Dies obwohl Eclipse die Bibliotheken inklusiv der Sourcen schon kennt. Jegliche Versuche mit einem Jar oder Zip Konstrukt scheiterten. Zu guter Letzt wurde ein Ansatz verfolgt, der nicht perfekt ist, jedoch funktioniert. Man suchte das Jar in der eigenen Workspace und nahm das erste Resultat, dieses konnte man dem Editor zum Öffnen übergeben, da es sich beim Resultat um ein IJavaElement handelte.

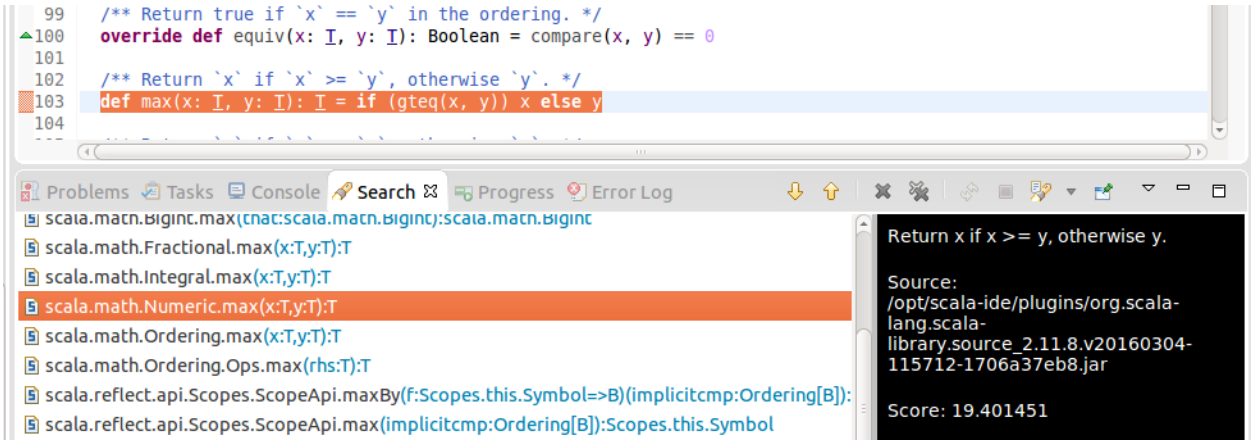


Abbildung 13: Markierung im Sourcecode

8.6.1.5 Scaps QuickFix

Bei der Besprechung am 16.03.2016 wurde vereinbart, dass der Quickfix einen Aufruf der Scaps Suche vorschlägt. Danach müssen aus dem Kontext die Informationen zusammengetragen werden, um die Suchmaschine damit befüllen. Die verwendete Query wird dabei schon im Quickfix angezeigt. Der Scaps QuickFix baut auf dem QuickAssist von der ScalaIDE auf. Das Verhalten ist sehr ähnlich zu dem QuickFix «create method», deshalb orientiert sich der Scaps QuickFix an diesem. Wenn der QuickFix angewendet wird, speichert der QuickFix die Suchanfrage als letzte ausgeführte Suchanfrage ab und öffnet den Suchdialog, dieser wählt dann die letzte Suchanfrage aus und fügt diese in das Suchtextfeld ein.

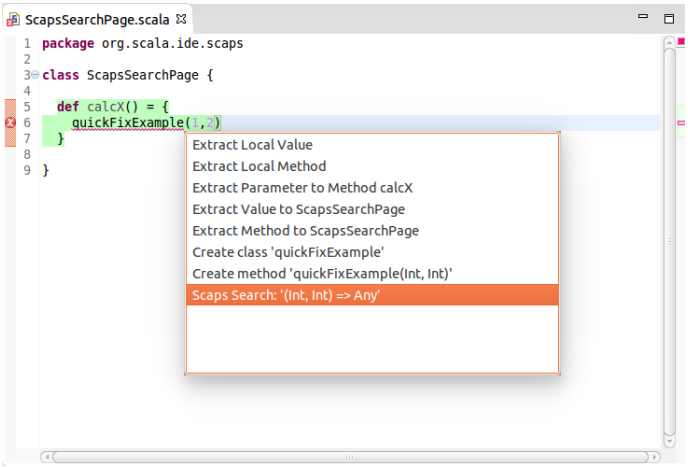


Abbildung 14:Screenshot Scaps QuickFix

Probleme

Schon während der Entwicklung des Prototypes wurden mehrere Versuche gemacht, einen Scaps Assistant zu entwickeln. Jegliche Versuche scheiterten in der Elaboration-Phase. Nachdem in der Construction-Phase die Integration und die Kernfunktion implementiert wurden, fiel der Fokus wieder auf den Scaps Quick Assistant. Verschiedene Ansätze wurden verfolgt, bis letztendlich eine Lösung in der Scala-IDE funktionierte. Durch eine Erweiterung des Scala Quickfix durch jemand Anderen, funktionierte der Quickfix des JDTs nicht mehr. Deshalb wurde auf das Scala Quickfix ausgewichen.
Die nächste Frage war, wie die Informationen aus dem Kontext geladen werden und was der genaue Vorschlag für den Entwickler sein soll. Bevor wir einen eigenen Algorithmus entwickelten, haben wir ein bewährtes Konzept gefunden. Wenn man in Eclipse eine Funktion mit Argumenten schreibt, die nicht existiert, gibt es eine Fehlermeldung. Beim Aufruf des Quickfix-Menüs wird dem Entwickler vorgeschlagen, eine neue Methode mit dem angegebenen Namen und Parametern zu erstellen, der Rückgabetyp wird dabei nicht ermittelt. Dies wäre auch eine Erweiterung für die Scala IDE.

8.7 Eclipse Architektur

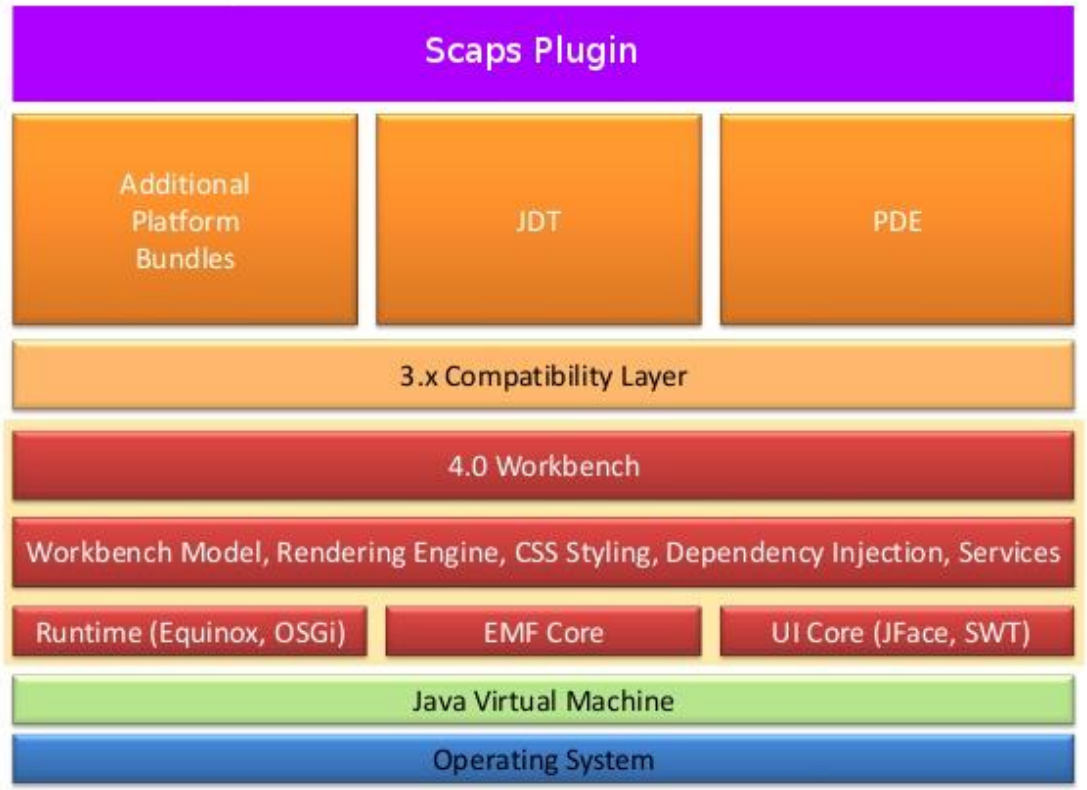


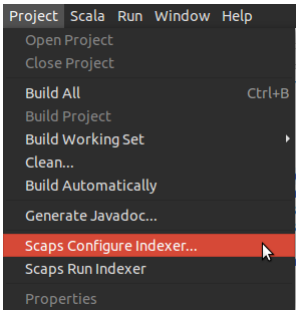
Abbildung 15: Eclipse Komponenten

Eine Eclipse Applikation besteht aus mehreren Eclipse Komponenten. Eine Software-Komponente in Eclipse nennt man ein Plugin. Die Eclipse Plattform ermöglicht es dem Entwickler, die Eclipse-IDE mit zusätzlichen Funktionalitäten über Plugins zu erweitern. Dies kann er über Extension Points erreichen, indem er im Plugin.xml die entsprechenden Zuweisungen macht.

Eclipse verwendet die Runtime-Anwendung, basierend auf der OSGi-Spezifikation. Eine Softwarekomponente in OSGi nennt man ein Bundle. Ein OSGi-Bundle ist auch ein Eclipse-Plugin. Beide Begriffe können austauschbar verwendet werden. Zum Beispiel kann ein neues Plugin neue Einträge im Toolbar-Menü erstellen.

Nachfolgend werden einige Eclipse Erweiterungen des Scaps Plugins erläutert.

8.7.1 Scaps Indexer Menu



Diese Extension wurde verwendet, um die Menüeinträge “Indexer konfigurieren” und “Indexer ausführen” darzustellen. Im Handler wird dann die Logik abgewickelt.

Abbildung 16: Screenshot Scaps Indexer Menu

8.7.1.1 Extension Points

Commands:
Definiert eine abstrakte Aktion, welche für die Zuweisungen benötigt wird.
Handlers:
In den Handlers definiert man welche Klasse ausgeführt werden muss, beim Betätigen eines Commands.
Menus:
Ein Command kann in einem Menu angezeigt werden.

8.7.1.2 Plugin.xml

```
<extension point="org.eclipse.ui.commands">
        <category
            name="Scaps Category"
            id="scaps.eclipse.ui.view.commands.category" />
        <command
            name="Scaps Run Indexer"
            categoryId="scaps.eclipse.ui.view.commands.category"
            id="scaps.eclipse.ui.view.commands.indexer.run" />
        ...
</extension>

<extension point="org.eclipse.ui.handlers">
        <handler
            commandId="scaps.eclipse.ui.view.commands.indexer.run"
            class="scaps.eclipse.ui.view.handlers.ScapsRunIndexerHandler" />
        ...
</extension>

<extension point="org.eclipse.ui.menus">
        <menuContribution locationURI="menu:project?after=additions">
                <separator name="separator-id" visible="true" />
                <command commandId="scaps.eclipse.ui.view.commands.indexer.configure" />
                <command commandId="scaps.eclipse.ui.view.commands.indexer.run" />
        </menuContribution>
</extension>
```

8.7.1.3 Sourcecode Ausschnitt

```
/* This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, You can obtain one at http://mozilla.org/MPL/2.0/. */

package scaps.eclipse.ui.view.handlers

import org.eclipse.core.commands.AbstractHandler
import org.eclipse.core.commands.ExecutionEvent
import scaps.eclipse.ui.handlers.IndexUCHandler
import org.eclipse.ui.handlers.HandlerUtil

class ScapsConfigureIndexerHandler extends AbstractHandler {

  def execute(event: ExecutionEvent): Object = {
    val window = HandlerUtil.getActiveWorkbenchWindowChecked(event)
    IndexUCHandler().configureIndexer(window)
    null
  }

}
```

8.7.2 Scaps Working Set

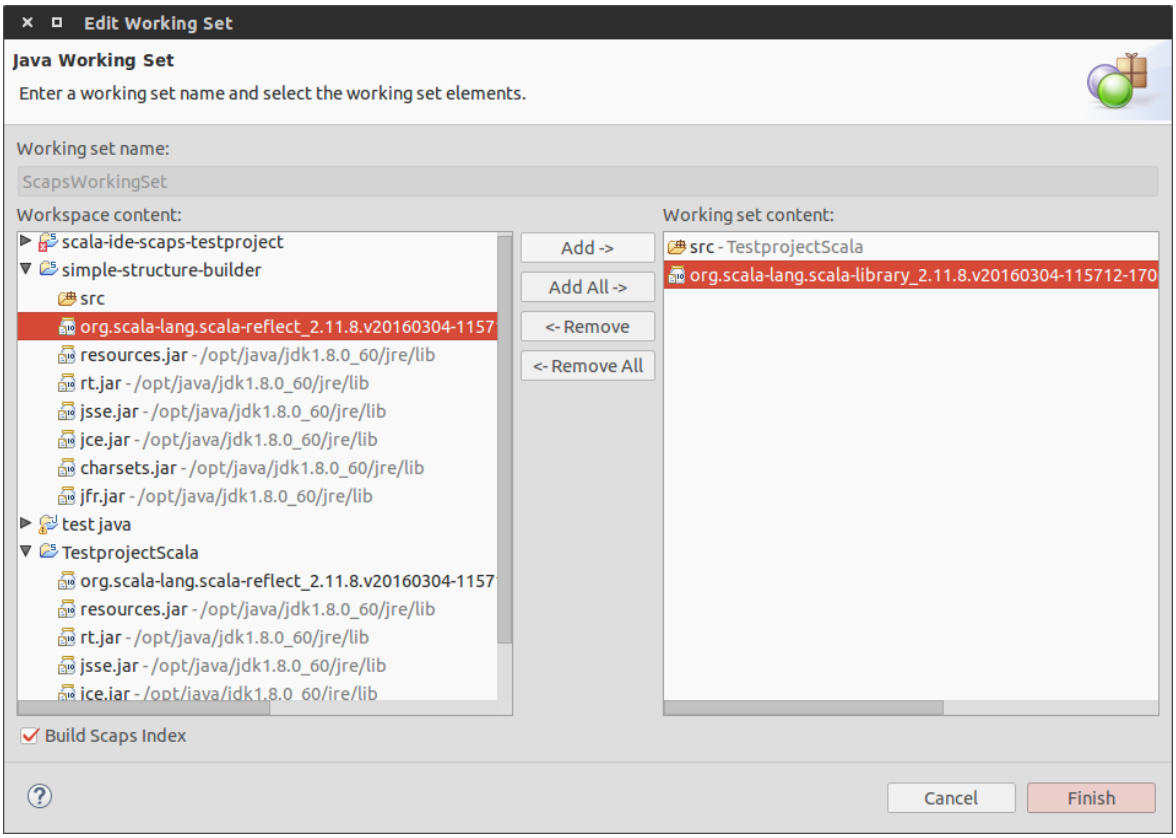


Abbildung 17: Screenshot WorkingSET

8.7.2.1 Extension Points

WorkingSets:
Mit dieser Extension wurde eine neue WorkingSetPage umgesetzt, damit der Entwickler die benötigten Ressourcen zum Indexieren auswählen kann.

8.7.2.2 Plugin.xml

```
<extension point="org.eclipse.ui.workingSets">
        <workingSet
                id="scaps.eclipse.ui.view.workingset.ScapsWorkingSetPage"
                name="Scaps Working Set"
                description="Contains projects and libraries witch will be indexed."
                icon="icons/resworkset.gif"
                pageClass="scaps.eclipse.ui.view.workingset.ScapsWorkingSetPage"
                updaterClass=
                "org.eclipse.jdt.internal.ui.workingsets.JavaWorkingSetUpdater"
                elementAdapterClass=
                "org.eclipse.jdt.internal.ui.workingsets.JavaWorkingSetElementAdapter">
        </workingSet>
</extension>
```

Extension point: Wird benötigt um eine eigene WorkingSetPage einzubinden
PageClass: der Pfad zu der neuen WorkingSetPage

8.7.2.3 Sourcecode Ausschnitt

```
/* This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, You can obtain one at http://mozilla.org/MPL/2.0/. */

package scaps.eclipse.ui.view.workingset

import org.eclipse.jdt.internal.ui.workingsets.JavaWorkingSetPage
import org.eclipse.jface.viewers.TreeViewer
import org.eclipse.swt.widgets.Composite
import org.eclipse.swt.widgets.Text
import org.eclipse.ui.PlatformUI
import scaps.eclipse.ui.handlers.IndexUCHandler
import org.eclipse.swt.SWT
import org.eclipse.swt.widgets.Button
import org.eclipse.swt.layout.GridData

class ScapsWorkingSetPage extends JavaWorkingSetPage {

  var performBuildCheckbox: Button = _

  override def createControl(composite: Composite): Unit = {
    super.createControl(composite)
    super.setTitle("Scaps Configure Indexer")
    setDescription("Select resources for index")
    // get the working set name text control and disable it
    …
  }
  override def configureTree(tree: TreeViewer): Unit = {
    super.configureTree(tree)
    tree.setContentProvider(new ScapsWorkingSetPageContentProvider())
  }

  override def finish(): Unit = {
    super.finish
    if (performBuildCheckbox.getSelection) {
      IndexUCHandler().runIndexer(
                PlatformUI.getWorkbench.getActiveWorkbenchWindow)
    }
  }

}
```

8.7.3 Search Menu

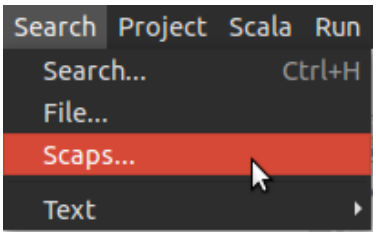


Abbildung 18: Screenshot SearchMenu

Der Search Menu Eintrag dient dazu, den Suchdialog zu öffnen.

8.7.3.1 Extension Points

ActionSet
Das ActionSet ist eigentlich deprecated, jedoch wurde das Suchmenü mit dem ActionSet realisiert, daher können Erweiterungen daran auch nur mit dem ActionSet gemacht werden. Mit dem ActionSet wurde ein Menüeintrag für die Scaps Suche implementiert.

8.7.3.2 Plugin.xml

```
<extension point="org.eclipse.ui.actionSets">
        <actionSet
                description="Scaps Search Action"
                id="scaps.eclipse.actionSet"
                label="Scaps Search Actions"
                visible="true">
                <menu
                        label="Search"
                        path="navigate"
                        id="org.eclipse.search.menu">
                        <groupMarker name="internalDialogGroup"/>
                        <groupMarker name="dialogGroup"/>
                        <separator name="fileSearchContextMenuActionsGroup"/>
                        <separator name="contextMenuActionsGroup"/>
                        <separator name="occurencesActionsGroup"/>
                        <separator name="extraSearchGroup"/>
                </menu>
                <action
                        class="scaps.eclipse.ui.view.actions.ScapsSearchAction"
                        enablesFor="*"
                        id="ScapsEclipse.OpenScapsSearch"
                        label="Scaps..."
                        menubarPath="org.eclipse.search.menu/dialogGroup"
                        style="push">
                </action>
        </actionSet>
</extension>
```

Extension point: Wird für einen Eintrag im Suchmenü benötigt

Action class: Diese Klasse beinhaltet die Logik bei der Auswahl für die Scaps Suche
MenubarPath: Zeigt an, dass dieser Eintrag am Schluss erfolgen muss

8.7.3.3 Sourcecode Ausschnitt

```
/* This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, You can obtain one at http://mozilla.org/MPL/2.0/. */

package scaps.eclipse.ui.view.actions

import org.eclipse.jface.action.IAction
import org.eclipse.jface.viewers.ISelection
import org.eclipse.ui.IWorkbenchWindow
import org.eclipse.ui.IWorkbenchWindowActionDelegate

import scaps.eclipse.ui.handlers.SearchUCHandler

class ScapsSearchAction extends IWorkbenchWindowActionDelegate {

  private var window: IWorkbenchWindow = _

  def init(window: IWorkbenchWindow): Unit = {
    this.window = window
  }

  def run(action: IAction): Unit = {
    if (window.getActivePage == null) {
      return
    }
    SearchUCHandler().openSearchDialog(window)

  }

  def selectionChanged(action: IAction, selection: ISelection): Unit = {}

  def dispose(): Unit = {
    window = null
  }

}
```

8.7.4 Search Page

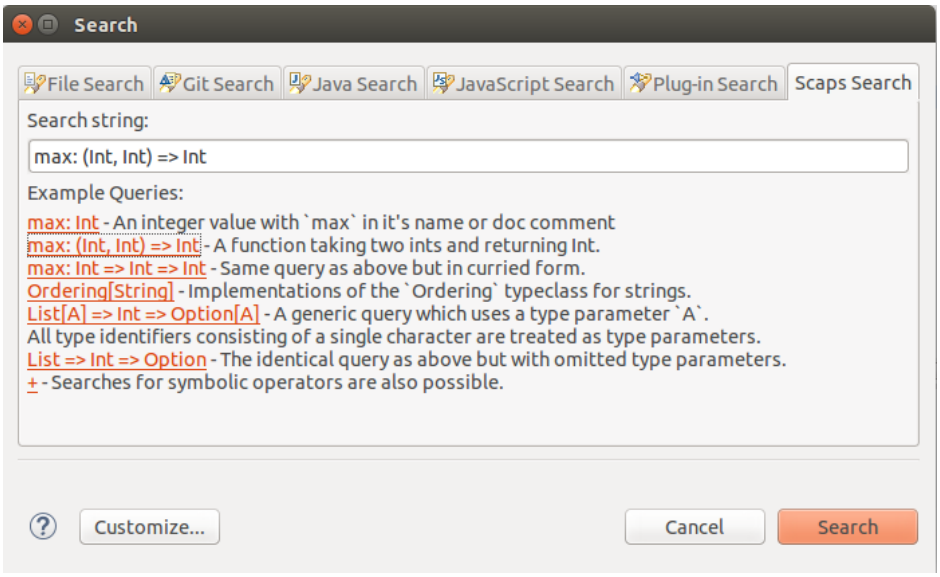


Abbildung 19: Screenshot SearchPage

8.7.4.1 Extension Points

SearchPages: Diese Extension wurde verwendet, um die Scaps Suchmaske zu entwickeln.

8.7.4.2 Plugin.xml

```
<extension point="org.eclipse.search.searchPages">
        <page id="scaps.eclipse.ui.view.search.ScapsSearchPage"
                label="Scaps Search"
                sizeHint="460,160"
                extensions="java:90, jav:90"
                showScopeSection="false"
                canSearchEnclosingProjects="true"
                class="scaps.eclipse.ui.view.search.ScapsSearchPage">
        </page>
</extension>
```

Extension point: Erzeugt einen eigenen Reiter für die Suche
Class: Pfad zur Sourceklasse der Suchmaske

8.7.4.3 Sourcecode Ausschnitt

```
/* This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, You can obtain one at http://mozilla.org/MPL/2.0/. */

package scaps.eclipse.ui.view.search

import org.eclipse.jface.dialogs.Dialog
import org.eclipse.jface.dialogs.DialogPage
import org.eclipse.search.ui.ISearchPage
import org.eclipse.search.ui.ISearchPageContainer
import org.eclipse.swt.SWT
import org.eclipse.swt.events.SelectionAdapter
import org.eclipse.swt.events.SelectionEvent
import org.eclipse.swt.layout.GridData
import org.eclipse.swt.layout.GridLayout
import org.eclipse.swt.widgets.Composite
import org.eclipse.swt.widgets.Label
import org.eclipse.swt.widgets.Link
import org.eclipse.swt.widgets.Text

import com.typesafe.scalalogging.StrictLogging

import scaps.eclipse.ui.handlers.SearchUCHandler
import scaps.eclipse.core.services.ScapsService

class ScapsSearchPage extends DialogPage with ISearchPage with
StrictLogging {
  def createControl(parent: Composite): Unit = {
    …
  }

  def performAction: Boolean = {
    SearchUCHandler().search(inputText.getText)
    true
  }

  def setContainer(container: ISearchPageContainer): Unit = {}
}
```

8.7.5 Search Result Page

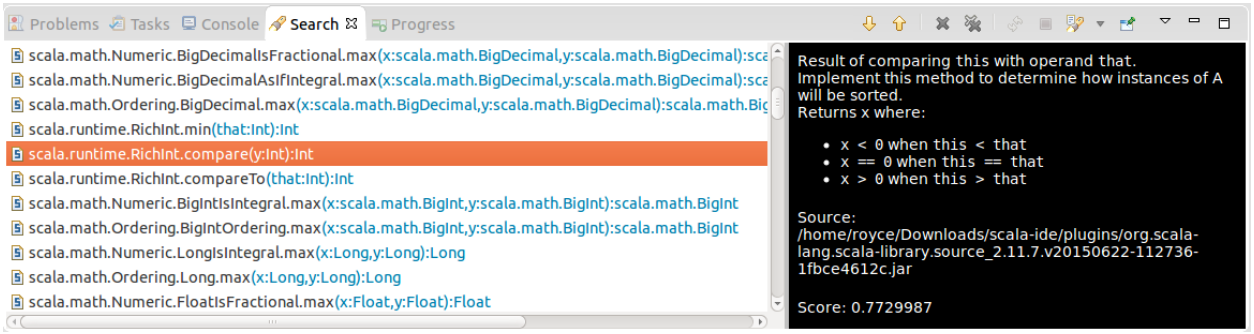


Abbildung 20: Screenshot Search Result Page

8.7.5.1 Extension Points

SearchResultViewPages: Diese Extension wurde verwendet, um die Resultate der Scapssuche darzustellen.

8.7.5.2 Plugin.xml

```
<extension point="org.eclipse.search.searchResultViewPages">
        <viewPage
                id="scaps.eclipse.ui.view.search.ScapsSearchResultPage"
                searchResultClass="scaps.eclipse.ui.search.ScapsSearchResult"
                class="scaps.eclipse.ui.view.search.ScapsSearchResultPage" />
</extension>
```

Extension point: Wird verwendet für eigene Suchresultat View
SearchResultClass: Gibt den (Klassen-)Typ der Resultate an (zwingend notwendig)
Class: Pfad zur Viewklasse

8.7.5.3 Sourcecode

https://github.com/flomerz/scala-ide-scaps/blob/master/scala-ide-scaps-plugin/src/main/scala/scaps/eclipse/ui/view/search/ScapsSearchResultPage.scala

8.7.6 Scaps QuickFix

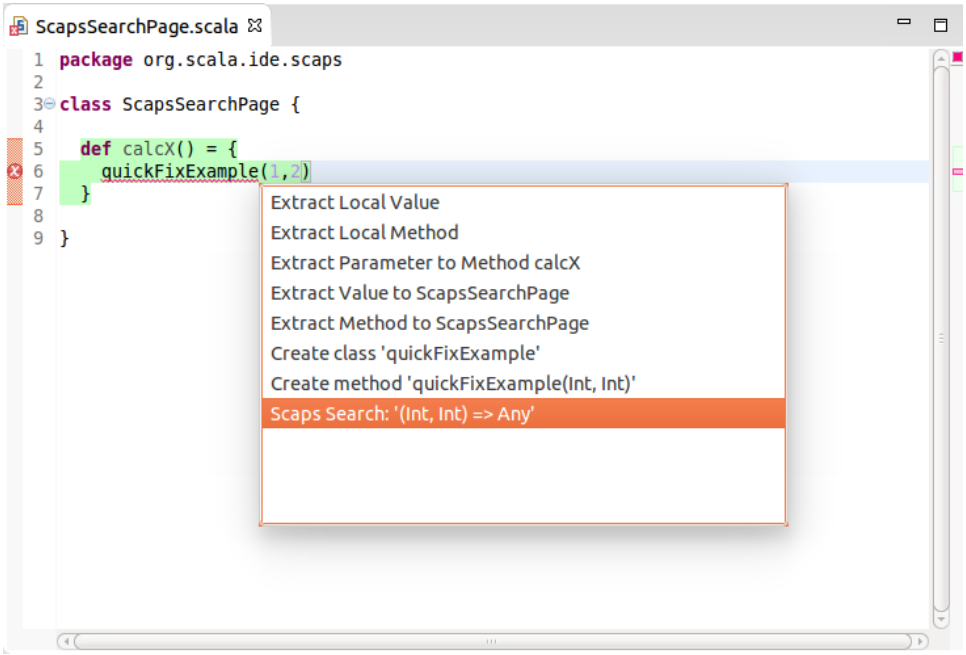


Abbildung 21: Screenshot Scaps QuickFix

8.7.6.1 Extension Points

QuickAssists: Diese Extension wurde verwendet, um dem Entwickler einen Suchquery im Editor vorzuschlagen.

8.7.6.2 Plugin.xml

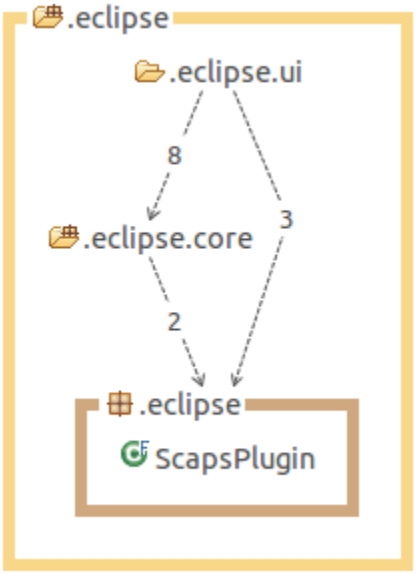
```
<extension point="org.scala-ide.sdt.core.quickAssists">
        <quickAssist
                class="scaps.eclipse.ui.view.quickfix.ScapsQuickAssist"
                id="scaps.eclipse.ui.view.quickfix.quickAssist" />
</extension>
```

Um beim Quickfix eine schnellere Antwort zu leisten, wird im Quickfix nur ein Queryvorschlag gemacht. Der Vorteil besteht darin, dass der Entwickler den Query noch anpassen kann und so erst später die Suche ausgeführt werden muss. Dieser Zwischenschritt garantiert eine schnelle Querybildung und gewährleistet, dass die Antwortzeit maximal 1 Sekunde dauert. Bei einer gleichzeitigen Suchanfrage wäre diese nichtfunktionale Anforderung nicht einzuhalten.

8.7.6.3 Sourcecode

https://github.com/flomerz/scala-ide-scaps/blob/master/scala-ide-scaps-plugin/src/main/scala/scaps/eclipse/ui/view/quickfix/ScapsQuickAssist.scala

8.8 Software Architektur



Die Softwarearchitektur für das Scaps Plugin baut auf den zwei Schichten UI und Core auf. Bei der Konzeption wurde vor allem Wert auf Anpassbarkeit gelegt.

8.8.1 UI

Die UI Schicht bildet die Schnittstelle zum Benutzer. Hier werden Benutzereingaben entgegengenommen, die notwendigen Kontext Informationen besorgt und zur Core Schicht weitergereicht. Diese Schicht ist aufgeteilt in drei Packages.

8.8.1.1 View

In dem View Package wird das Grafische abgewickelt.

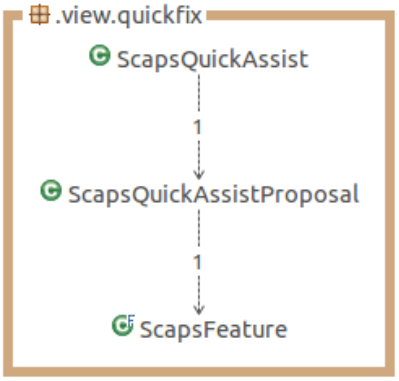
Actions

Hier werden die Eclipse Actions verarbeitet. Diese sind deprecated, deshalb wurden diese nur wenn es notwendig war verwendet.

Eclipse Handlers

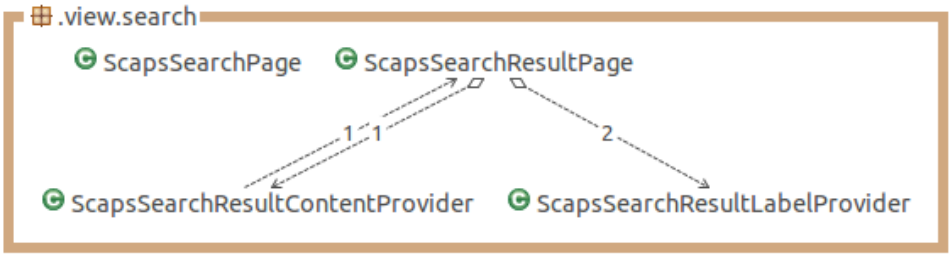
Die Eclipse Handlers sind die neuen Actions. Hier werden die Benutzerinteraktionen auf Commands verarbeitet.

Quickfix



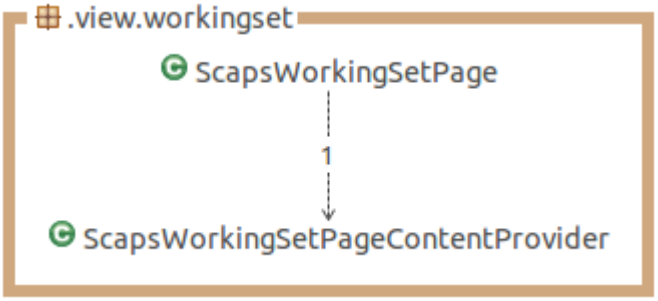
Im Quickfix Package wird alles bezüglich dem Quickfix und dem Vorschlag abgewickelt. Der ScapsQuickAssistProposal sucht sich die Typen für die Suchanfrage zusammen.

Search



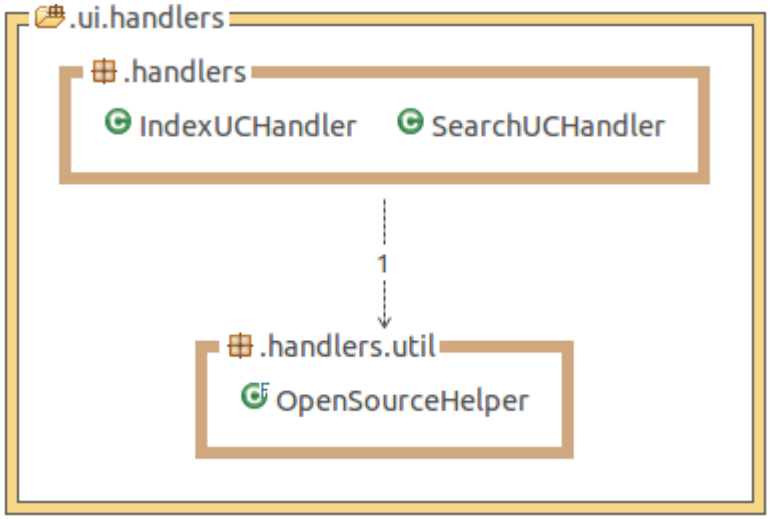
In dem Search Package ist die Suchmaske und die Resultview mit den zugehörigen Providern.

WorkingSet



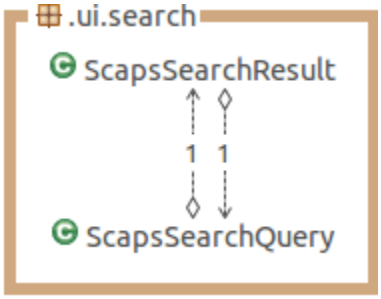
Im WorkingSet Package wird der Scaps WorkingSet Wizard mit den benötigten Anpassungen gebaut.

8.8.1.2 Use Case Handler



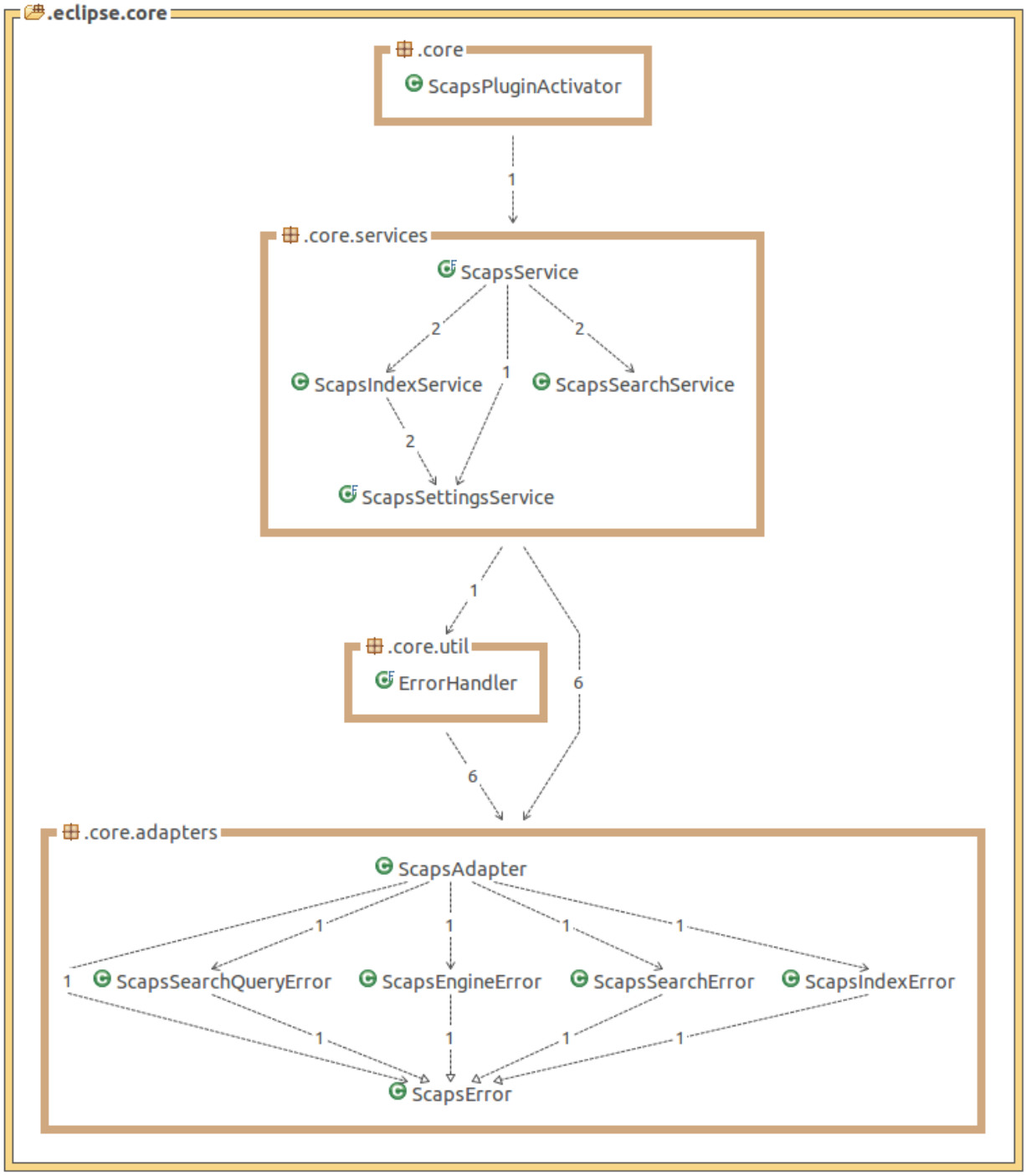
Die UCHandlers sind nach dem Use Case Controller Pattern umgesetzt worden. Sie nehmen einige Kontext Informationen von den Actions entgegen und beschaffen sich alle weiteren Informationen, die für diese Aufgabe notwendig sind, so zum Beispiel Pfade zu den Libraries. Damit der OpenSourceHelper die Sourcen öffnen kann, muss eine *.class Datei aufgerufen werden. Um das zu erreichen, wird die Dateiendung *.scala mit *.class ersetzt. Sollten später auch *.java Dateien geöffnet werden müssen, muss dieser entsprechen erweitert werden.

8.8.1.3 Search



In diesem Package werden Konvertierungen vorgenommen, so dass die Suchanfragen und Suchresultate mit den Eclipse Komponenten verwendet werden können.

8.8.2 Core

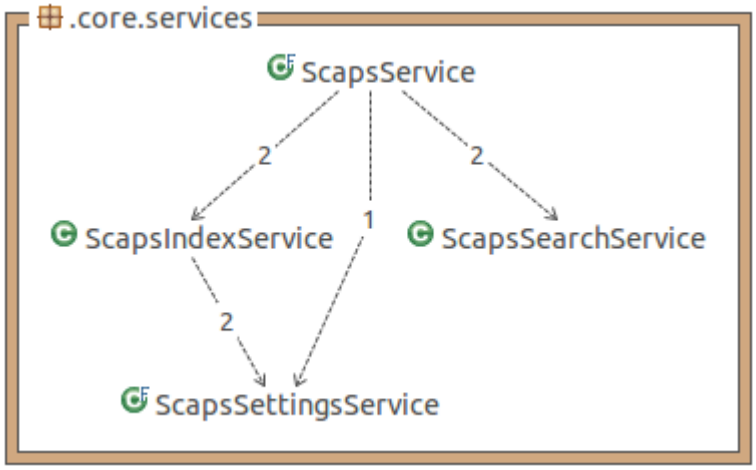


Die Core Schicht nimmt nur Aufrufe von der UI Schicht entgegen. Sie ist zuständig für die eigentliche Ausführung einer Aktion, wie zum Beispiel die Ausführung einer Indexierung oder Suche.

8.8.2.1 Core

Er beinhaltet einen Plugin Activator. Der Activator wird für das OSGi-Bundle verwendet.

Services

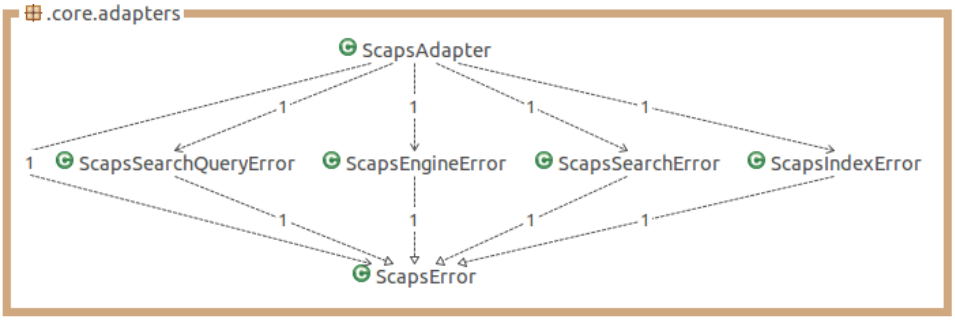


Die Services sind dafür zuständig, dass zeitintensive Jobs im Hintergrund ablaufen. Sie sind nach dem Service Pattern aufgebaut und sind für einen Teilbereich zuständig.

Util

Der ErrorHandler erwartet einen ScapsError. Diesen kann er verarbeiten und zum Beispiel eine grafische Fehlermeldung anzeigen und den StackTrace in der ErrorLog View darstellen.

Adapters



Alle Zugriffe auf Scaps werden über diesen Adapter gemacht. Das Handling der Errors, die aus Scaps kommen, werden hier gehandelt. Hier wurde das Adapter Pattern angewendet, um die Methoden in die gewünschte Form zu bringen.

Eclipse

Hier sind einige Einstellungen, wie der Pfad für die Indexierung definiert.

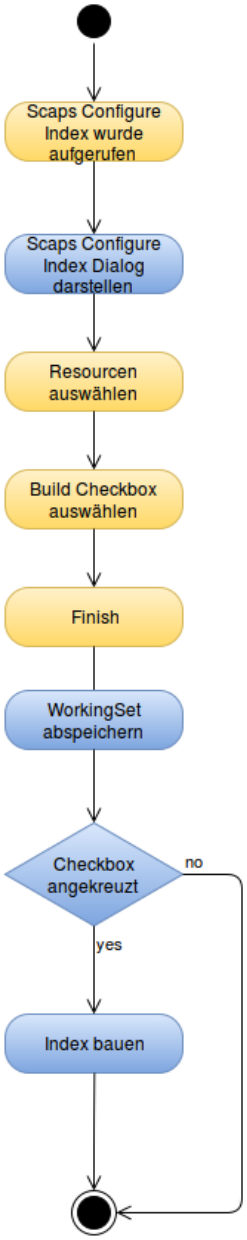
8.9 Indexiervorgang

Der Indexiervorgang besteht aus zwei Teilen. Die beiden Teile sind voneinander abhängig.

8.9.1 Index-Ressourcen auswählen

Das Diagramm zeigt den Ablauf des Use Cases Index-Ressourcen auswählen. Das Diagramm enthält zwei Farbblöcke, die gelben Blöcke repräsentieren die Aktionen des Entwicklers, während die blauen Blöcke die Systemaktionen repräsentieren.

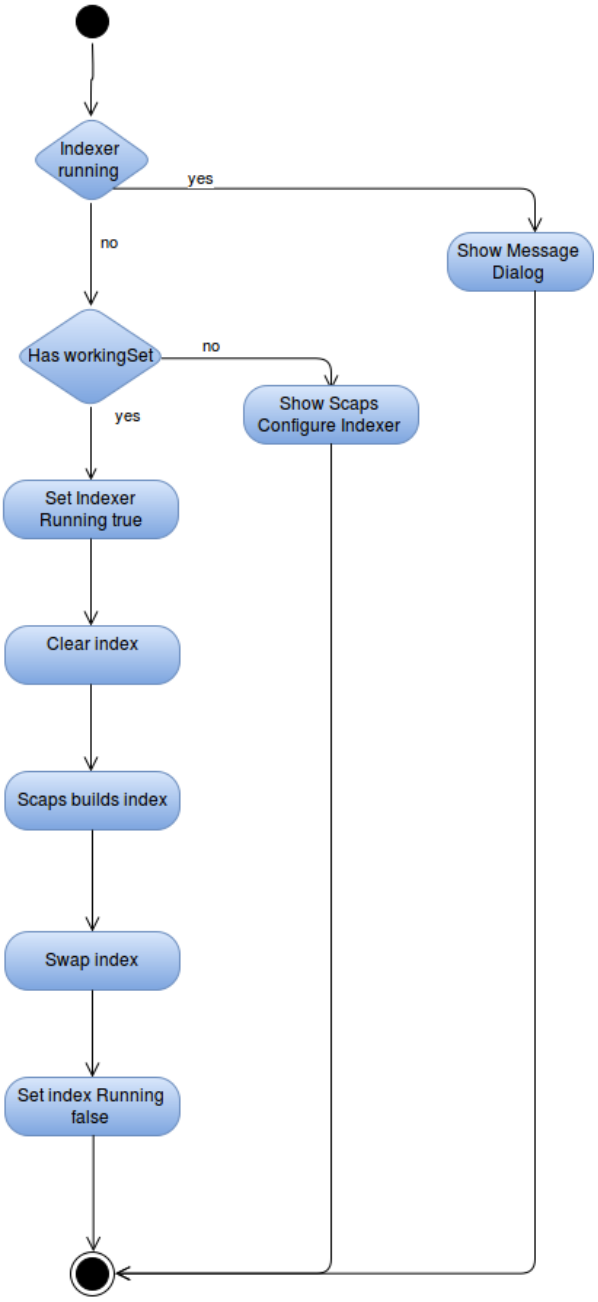
1. Der Entwickler wählt im Menü Scaps Configure Index aus.
2. Ein Dialog öffnet sich und zeigt alle Projekte, beziehungsweise Ressourcen, im Tree-Modus an.
3. Der Entwickler sucht sich die Ressourcen aus, welche er dem Index hinzufügen möchte.
4. Der Entwickler kann gleich nach der Auswahl der Ressourcen den Index builden lassen, dazu muss er die Checkbox aktivieren.
5. Der Entwickler beendet die Auswahl der Ressourcen.
6. Falls kein Scaps Working Set existiert, wird ein neues angelegt und die ausgewählten Ressourcen darin referenziert. Falls bereits ein Scaps Working Set existiert, werden die Änderungen darin abgespeichert.
7. Falls die Checkbox angekreuzt ist, Punkt 8, ansonsten Punkt 9.
8. Use Case Index erstellen.
9. Prozessende



8.9.2 Index erstellen

Sobald die Indexierung ausgelöst wurde, sind keine Interaktion von Seiten des Entwicklers mehr nötig.

1. Das System prüft, ob ein Indexer schon läuft, falls ja, Punkt 9, falls nein, Punkt 2.
2. Das System prüft, ob es schon ein Scaps Working Set hat, falls ja, Punkt 3, falls nein, Punkt 10.
3. Ein Flag wird gesetzt, dass der Indexer schon läuft.
4. Der passive Index wird geleert. Es werden zwei Indexe verwendet, sodass während des Indexiervorgangs ein Index immer noch für die Suche verwendet werden kann. Dies stellt die nichtfunktionale Anforderung Zuverlässigkeit sicher.
5. Der passive Index wird gebaut. Der Index wird als Background-Prozess gebaut, so kann Eclipse weiterhin verwendet werden. Dies stellt die nichtfunktionale Anforderung Zuverlässigkeit sicher.
6. Der aktive und passive Index wird ausgetauscht.
7. Das Flag "Indexer läuft" wird geändert zu "läuft nicht".
8. Prozessende
9. Ein Message Dialog mit der Nachricht "Index wird bereits gebaut" wird gezeigt. Weiter zu Punkt 8.



10. Punkt 8 + der Use Case Index-Ressourcen wird aufgerufen.

8.10 Framework & Tools

8.10.1 e4 vs. PDE

Zur Auswahl für die Entwicklung von Plugins standen e4 und PDE zur Verfügung. E4 steht für Eclipse 4, was durch einige Designänderungen in Eclipse entstand und ist ab Eclipse Version 4 verfügbar. PDE steht für Plugin Development Environment, dies ist die ältere Schnittstelle, kann aber ebenfalls auf der neusten Eclipse Umgebung verwendet werden, da Eclipse eine 3.x Kompatibilität Schicht hat. Die JDT Java Development Tools und die ScalaIDE verwenden PDE für die Eclipse Erweiterungen, deshalb haben wir uns für PDE entschieden, damit wir unsere Erweiterungen auf diesen aufbauen können.

8.10.2 Eclipse & Java

Zum Einstieg wurde in der Elaboration Phase mit Eclipse und Java gearbeitet. Dies bescherte uns den Vorteil, dass unsere Erfahrungen in Java und Eclipse ins Projekt einbezogen werden konnten. Viele Codebeispiele wurden mit Java umgesetzt. Nach der Elaboration Phase wurden diese dann ohne grosse Probleme in Scala Code umgeschrieben.

8.10.3 Diagramme

Astah wurde verwendet, um die Use Case Diagramme und Komponenten Diagramme zu entwerfen.
Das Plugin Stan wurde in Eclipse für die Qualitätssicherstellung und für Auszüge in der Dokumentation verwendet.

8.10.4 Wireframesketcher

Mit diesem Tool haben wir das Design der Screens erstellt, nach denen wir in der Construction vorgegangen sind.

8.10.5 Infrastruktur

Zur Grundinfrastruktur gehört die Scala IDE, Maven, Jenkins und die Programmiersprachen Scala, sowie Java für den Prototype.
Zur Kommunikation wurde Slack verwendet, welches auch an unser Projektmanagement-Tool Redmine, Git (https://github.com/flomerz/scala-ide-scaps) und an Jenkins angebunden wurde.

8.10.5.1 Maven

Das Projekt kann mit Maven gebaut werden. Das wird vor allem für Jenkins gebraucht.

Tycho Plugin

Wird benötigt um ein Eclipse Plugin mit Maven zu bauen.

```
<plugin>
	<groupId>org.eclipse.tycho</groupId>
	<artifactId>tycho-compiler-plugin</artifactId>
	<version>${tycho.version}</version>
	<configuration>
		<excludeResources>
			<excludeResource>**/*.scala</excludeResource>
		</excludeResources>
	</configuration>
</plugin>
```

Maven Dependency Plugin

Damit die transitiven Bibliotheken mit Sourcecode auch im Eclipse zur Verfügung stehen, wird dieses Plugin gebraucht.

```
<plugin>
	<groupId>org.apache.maven.plugins</groupId>
	<artifactId>maven-dependency-plugin</artifactId>
	<version>2.10</version>
	<executions>
		<execution>
			<id>copy-source-dependencies</id>
			<phase>validate</phase>
			<goals>
				<goal>copy-dependencies</goal>
			</goals>
			<configuration>

	<outputDirectory>${project.build.directory}/sources</outputDirectory>
				<overWriteReleases>false</overWriteReleases>
				<overWriteSnapshots>false</overWriteSnapshots>
				<overWriteIfNewer>true</overWriteIfNewer>
				<excludeGroupIds>p2.eclipse-plugin</excludeGroupIds>
				<classifier>sources</classifier>
			</configuration>
		</execution>
		<execution>
			<id>copy-binary-dependencies</id>
			<phase>validate</phase>
			<goals>
				<goal>copy-dependencies</goal>
			</goals>
			<configuration>

	<outputDirectory>${project.build.directory}/libs</outputDirectory>
				<overWriteReleases>false</overWriteReleases>
				<overWriteSnapshots>false</overWriteSnapshots>
				<overWriteIfNewer>true</overWriteIfNewer>
				<excludeGroupIds>p2.eclipse-plugin</excludeGroupIds>
			</configuration>
		</execution>
	</executions>
</plugin>
```

8.11 Testing

Die Applikation enthält wenige Tests zur Sicherstellung der Grundfunktionalitäten. Es wurden Unit und Integration Tests geschrieben. Wobei beide über Maven ausgeführt werden können. Die Integration Tests starten eine Eclipse Instanz, um die Tests im Kontext von Eclipse laufen zu lassen. Damit die Tests auch auf dem Jenkins CI laufen, wurde das Programm Xvfb verwendet, welches eine virtuelle Oberfläche erstellen kann, um die Eclipse Instanz zu öffnen. Die Tests sind sicherlich erweiterbar, jedoch macht das nur bis zu einem gewissen Grad Sinn, da die Applikation selbst keine grosse Logik beinhaltet. Zudem gestaltete sich die Testumgebung als sehr aufwändig. Es wurde viel Zeit investiert, um ein paar Grundfunktionalitäten testen zu können.

8.11.1 Unit

Für Unit Tests kommt das Mockito Framework zum Einsatz.
Hier ein Ausschnitt aus einem TestCase:

```
@RunWith(classOf[MockitoJUnitRunner])
class ScapsIndexServiceUnitTest {

  val scapsIndexService = new ScapsIndexService(mock(classOf[ScapsAdapter]))

  val mockedWorkingSet = mock(classOf[IWorkingSet])
  val mockedJavaProject = mock(classOf[IJavaProject])
  val mockedClassPathEntry = mock(classOf[IClasspathEntry])
  val mockedClassPathEntry2 = mock(classOf[IClasspathEntry])
  …

  @Before
  def setupRules() {

when(mockedJavaProject.getResolvedClasspath(anyBoolean)).thenReturn(Array(mockedClassPathEntry
, mockedClassPathEntry2))
    when(mockedClassPathEntry.getPath).thenReturn(mockedClassPath)
    when(mockedClassPathEntry2.getPath).thenReturn(mockedClassPath2)
    when(mockedClassPathEntry.getSourceAttachmentPath).thenReturn(mockedClassPath)
    …
  }

  @Test
  def testExtractOfIJavaProject {
    // setup
    when(mockedWorkingSet.getElements).thenReturn(
          Array(mockedJavaProject).asInstanceOf[Array[IAdaptable]])
    when(mockedJavaProject.getAllPackageFragmentRoots).thenReturn(
          Array(mockedPackageFragment, mockedPackageFragment2))

    // SUT
    val result = scapsIndexService.extractElements(mockedWorkingSet)

    // verify
    assertEquals(2, result._1.size)
    assertTrue(result._1.contains(classPath))
    assertTrue(result._1.contains(classPath2))
    …
  }
}
```

8.11.2 Integration

Für Integration Test wurde die ScalaIDE Testumgebung verwendet.:
http://scala-ide.org/docs/dev/testing/eclipse-tests.html
Hier ein Ausschnitt aus einem Integrationstest:

```
class ScapsAdapterIntegrationTest extends TestProjectSetup("simple-structure-builder") with
StrictLogging {

  @Test
  def testProjectIndexing: Unit = {
    val indexDir = ResourcesPlugin.getWorkspace.getRoot.getLocation.toOSString +
"/testing/index"
    logger.info(s"IndexDir: $indexDir")
    val scapsAdapter = new ScapsAdapter(indexDir)
    // setup
    addSourceFile(project)("Calculator.scala", """
      class Home {
        def plus(num1: Int, num2: Int): Int = num1 + num2
      }""")
    val javaProject = project.javaProject
    val classPath = extractClassPath(javaProject)
    val projectSourceFragmentRoots = javaProject.getAllPackageFragmentRoots.filter(_.getKind
== IPackageFragmentRoot.K_SOURCE).toList
    val compilationUnits = projectSourceFragmentRoots.flatMap(findSourceFiles)

    // SUT
    val indexResetResult = scapsAdapter.indexReset
    val indexResult = scapsAdapter.indexProject(classPath, compilationUnits)
    val indexFinalizeResult = scapsAdapter.indexFinalize
    val searchResult = scapsAdapter.search("Home")

    // verify
    assertTrue(indexResetResult.isRight)
    assertTrue(indexResult.isRight)
    assertTrue(indexFinalizeResult.isRight)
    assertTrue(searchResult.isRight)
  }
  ...
}
```

8.12 Bedienungsanleitung

As soon as the plugin is available online you can download it at the following update site: http://sinv-56022.edu.hsr.ch/jenkins/job/scala-ide-scaps/ws/scala-ide-scaps-updatesite/target/site.zip
Follow these instructions to use Scaps Search IDE Plugin:

8.12.1 Configure Resources

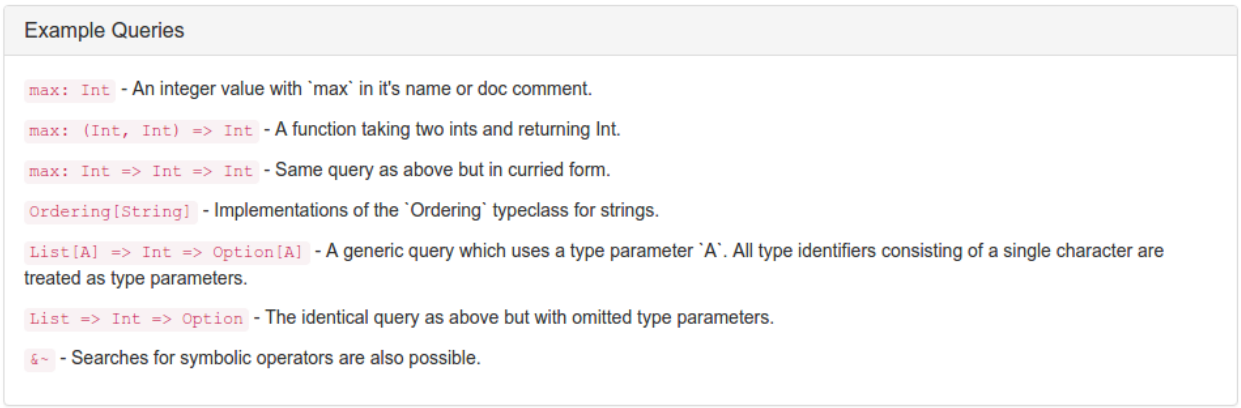
1. Go to Project menu
2. Select Scaps Configure Indexer
3. Select all resources you need for your search
4. Select the checkbox if you want to index the project now (this process can take some time)
5. Click on the finish button

8.12.2 Run Indexer

1. Go to Project
2. Select Scaps Run Indexer

8.12.3 Search

These are example Queries for the Scaps Search from the Webapplication.



8.12.4 Go to Search

1. Select Scaps
2. Enter your searchquery
3. Click search
4. Results will be shown in the result view
5. A simple click on one result will give you further information about the function
6. A double click opens the class where the function is defined

8.12.5 Tips and Tricks

- If you're in the editor, you can write a new method signature for which you are looking for. It should appear a problem marker because there is no such function. You can press now CTRL+1 and a quickfix dialog will pop up. At the bottom of all quick fixes there should be an entry of Scaps Search. By clicking on that the ScapsSearchPage will appear with a prefilled query based on the function you are looking for.
- You can directly use the functions displayed in the result view by CTRL+C and paste it in to the editor by CTRL+V

8.13 Schlussfolgerung

Das Produkt der Bachelorarbeit ist ein Plugin für die Scala IDE. Das Plugin ist ein erster Schritt, um den Suchprozess des Entwicklers zu verbessern. Folgende Ziele der Bachelorarbeit wurden erreicht:

- Setup der Entwicklungsumgebung mit Eclipse und der Scala IDE. Das Setup von Eclipse wurde in der Elaboration Phase für den Prototyp verwendet. Später wurde auf die Scala IDE gewechselt, der Wechsel brachte einige Probleme mit sich. Auch wenn die Scala IDE auf Eclipse aufbaut, ist die Benutzerführung nicht so ideal für einen Neueinsteiger in Scala.
- Integration der Suchmaschine in die IDE: Die Projektinformationen aus der IDE sollen genutzt werden, um den Index aufzubauen. Dieses Ziel wurde, mit Hilfe von den Use Case Index-Ressourcen auswählen und Index erstellen, abgedeckt.
- Entwickeln des Such UIs: Hier sind innovative Ideen gefragt, wie die Suche optimal in den Workflow eines Entwicklers eingebettet werden kann. Die Suche sollte, ähnlich wie Code Completion, direkt aus dem Editor gestartet werden können. Mit der Suchmaske und dem Quickfix werden dem Entwickler zwei gängige Lösungen zur Verfügung gestellt, welche dank den einfachen Benutzerführungen selbstverständlich sind.

Die Bachelorarbeit ermöglichte einen spannenden Einblick in die Entwicklungsumgebung Eclipse, die wir als Entwickler schon seit längerem benutzen. Ausserdem konnten wir Scala zum ersten Mal produktiv einsetzen.

Wir sind erfreut über den Ausgang dieser Bachelorarbeit und hoffen, dass dieses Plugin den Entwicklern eine Hilfe ist. Das Produkt kann als Grundlage für weitere Verbesserungen des Suchprozesses angesehen werden.

Folgende Punkte dienen als Ideen wie man das Plugin ausbauen könnte.

8.13.1 Scoreboost

Die eigenen Projektsourcen sollen in der Result View weiter oben angezeigt werden.

8.13.2 Kontinuierliche Indexierung

Der Indexer wird automatisch neu gebaut, wenn sich genügend Ressourcen geändert haben. Dies reduziert den Zeitaufwand beim Indexierprozess, da weniger Ressourcen für den Index durchsucht werden müssen.

8.13.3 Mehrere Indexe

Der Entwickler kann mehrere Indexe anlegen. Dadurch verkürzt sich der Zeitaufwand beim Indexierprozess.

8.13.4 Quickfix

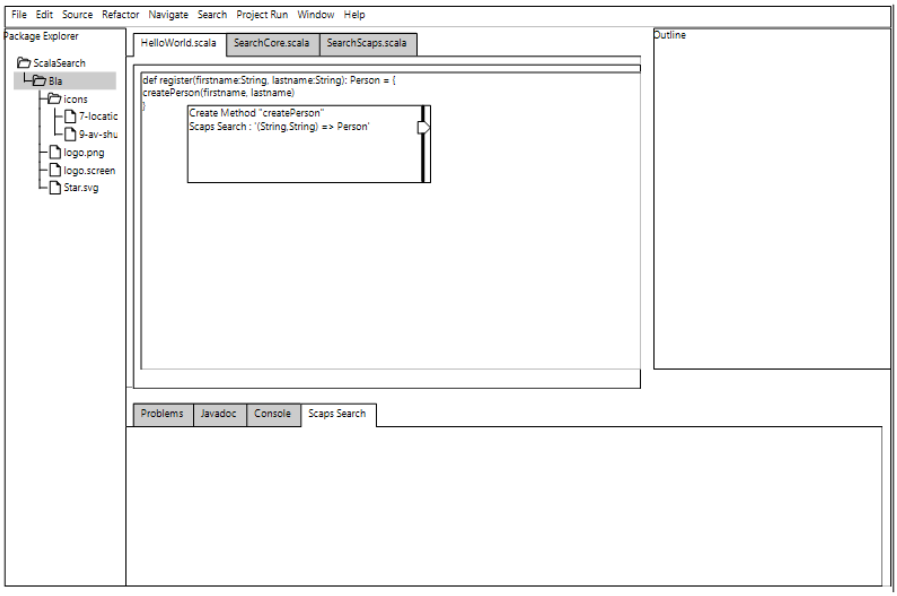


Abbildung 22: Wireframe QuickFix Future

Zurzeit wird bei einem Queryvorschlag als Rückgabetyp Any angegeben. In Zukunft wäre es für den Entwickler spannend, wenn das Plugin den Rückgabetyp schon erahnen würde. Ein Beispiel dazu sieht man im obigen Wireframe. In diesem Fall wird der Rückgabetyp aus der Methodensignatur erahnt.

8.13.5 Neuer Dialog

Folgendes Wireframe zeigt das Konzept des Dialoges.

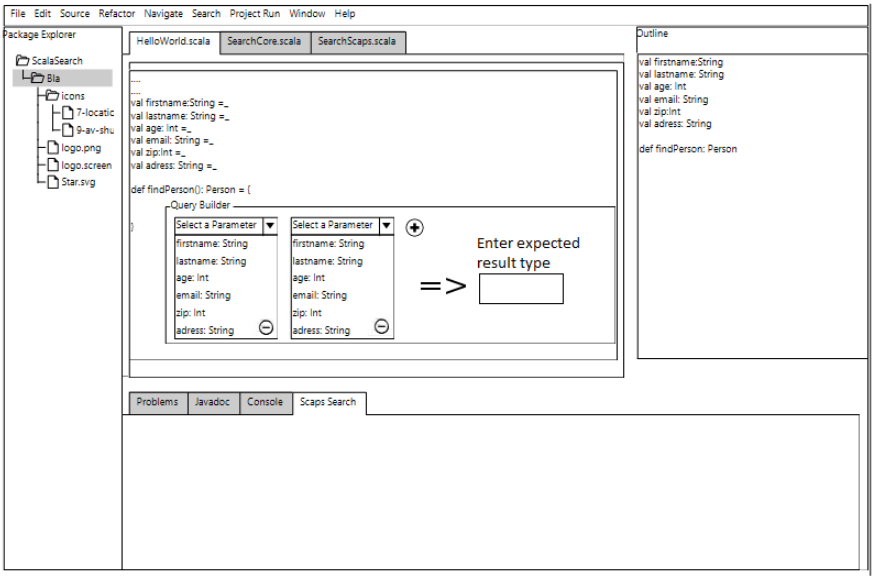


Abbildung 23: Wireframe ScapsAssist Future

Der Dialog soll dem Entwickler als Eingabehilfe für die Querybildung dienen. Dem Entwickler werden alle im Kontext vorhandenen Variablen angezeigt und der Entwickler kann auswählen, mit welchen Variablen die Query gebildet werden muss. Am Schluss kann noch der Rückgabetyp eingegeben werden. Falls die oben genannte Quickfix Erweiterung schon vorhanden ist, kann man hier auch schon den Rückgabentyp erahnen.

8.13.6 Java

Scaps wird um die Indexierung von Java Sourcen erweitert. Somit könnte dieses Plugin auch für Java-Entwickler von Vorteil sein.

8.14 Projektmanagement

Für das Projektmanagement wurde das Ticketingsystem Redmine verwendet. Es wurden wöchentlich Meetings abgehalten, um den Stand mitzuteilen und offene Fragen zu klären. Die wöchentlichen Meetings sind in Redmine protokolliert. Florian Merz und Royce Manavalan sind in der Rolle des Projektleiters und Entwicklers.

8.14.1 Projektplan

Das Projekt wird nach RUP durchgeführt. Die Gesamtprojektdauer beträgt 17 Wochen, dementsprechend wurde der Zeitplan gemacht. Der Start einer Iteration ist jeweils ein Mittwoch. Nach der Construction Phase wurde eine Reserve von zwei Wochen eingebaut, um allenfalls auftretende Verzögerungen abzufangen.

8.14.2 Phasen

Phase	Anfang	Dauer
Inception	24. Februar	1 Woche
Elaboration	02. März	6 Wochen
Construction	13. April	7 Wochen
Transition	15. Juni	2 Wochen

8.14.2.1 Inception

In dieser Phase liegt der Fokus darauf, verschiedene Ideen zu finden und zu evaluieren, in welche Richtung das Projekt zu leiten ist. Es sollte einen kleinen Einblick in die verschiedenen Möglichkeiten bieten.

8.14.2.2 Elaboration

Der Fokus in dieser Phase wird grösstenteils auf den Prototyp gesetzt, nebenbei werden Projektstrukturplan und die Wireframes erstellt, die dem Prototyp dienen.

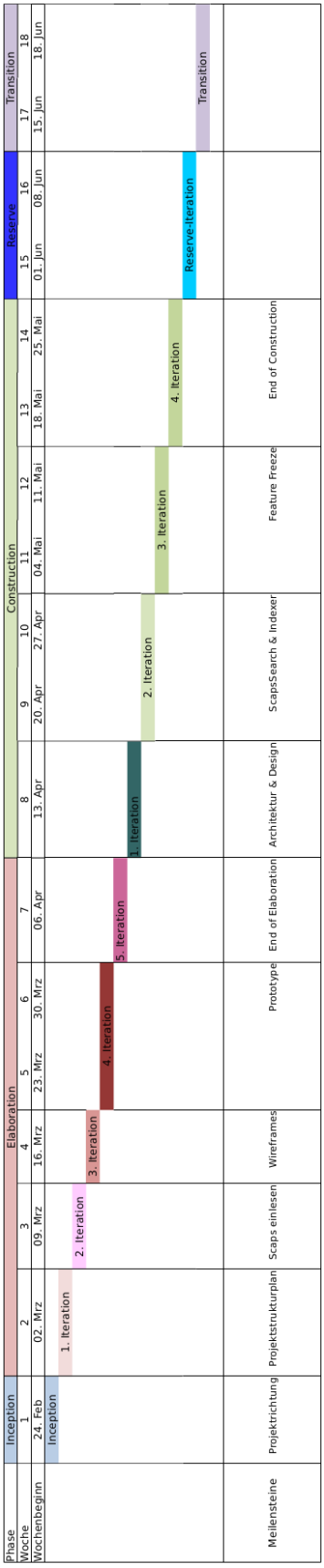
8.14.2.3 Construction

In dieser Phase wird das Konzept aus der Elaboration umgesetzt.

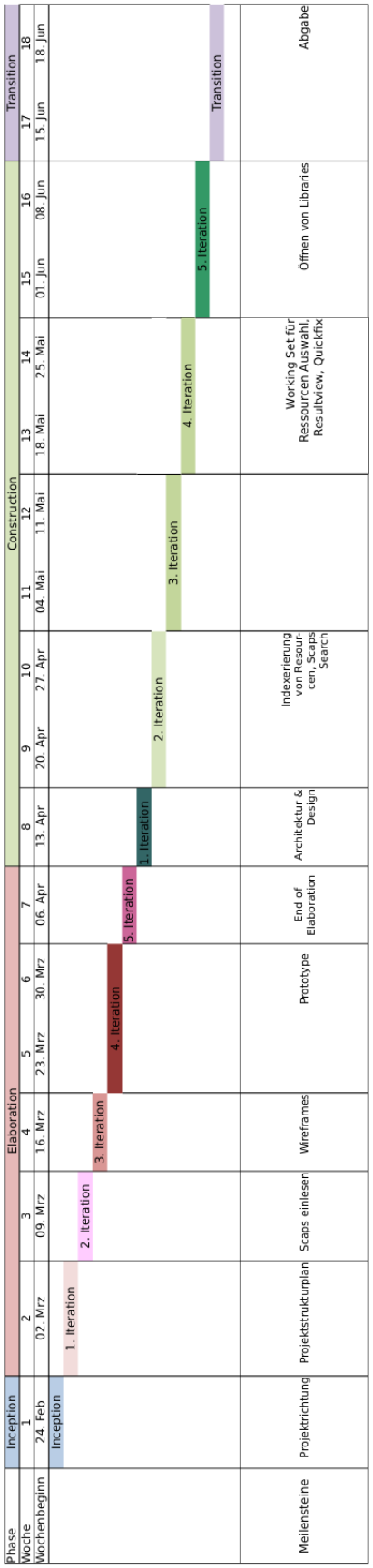
8.14.2.4 Transition

In der Transition wird die Abgabe vorbereitet und die Dokumentation, sowie das Poster und die Präsentation fertiggestellt.

Soll Plan



Ist Plan



8.14.3 Meilensteine

Um ein Tracking des Standes zu gewährleisten wurden folgende Meilensteine von uns festgelegt.

8.14.3.1 Projektrichtung – 02.03.2016

Schnell wurden wir uns einig, dass unser Fokus auf der Integration von Scaps mit einer guten Benutzeroberfläche liegt. Danach folgte die Idee des Quickfixes.

8.14.3.2 Projektstrukturplan – 09.03.2016

Der Projektstrukturplan diente uns als Grundlage für das Projektmanagement. Anhand von dieser Planung haben wir auch Redmine aufgesetzt.

8.14.3.3 Scaps einlesen – 16.03.2016

Dieser Meilenstein ist nicht messbar, jedoch muss eine gewisse Vorarbeit geleistet werden, bevor man sich in der Phase dem Problem widmet.

8.14.3.4 Wireframes – 23.03.2016

Während der Elaboration Phase wurden uns die Möglichkeiten von Eclipse bekannt. Die Wireframes dienen als Grundlage für die Entwicklung.

8.14.3.5 Prototyp – 30.03.2016

Am Ende der Elaboration Phase wurde ein Prototyp vorgestellt. Dieser konnte die Grundfunktionen von Scaps Indexieren und Suchen und hatte ein kleines GUI. Der Quickfix wurde nach einigen Versuchen liegen gelassen.

8.14.3.6 Architektur – 13.04.2016

Die Architektur wurde zusammen mit dem Prototyp gemacht. Die Architektur, die hier im Meilenstein beschrieben ist, dient als Grundlage für die Construction Phase. Während der Construction Phase wurde die Architektur erweitert.

8.14.3.7 Suchmaske & Indexer erledigt - 04.05.2016

Die Grundfunktionalitäten, also das Indexieren von Ressourcen und das Suchen nach Funktionen, wurde als grosser Meilenstein definiert. Leider trafen bei der Entwicklung die oben genannten Probleme ein, welche eine Verzögerung der kompletten Funktionalitäten mit sich brachten.

8.14.3.8 Quickfix - 25.05.2016

In dieser Iteration wurde der Use Case Index Ressourcen auswählen und die Resultview fertig implementiert.
Der Quickfix wurde als letzter Meilenstein in der Construction-Phase definiert. Er soll dem Entwickler helfen, eine Funktion während dem Programmieren zu finden. Dieser wurde nach Startschwierigkeiten nach Plan umgesetzt.

8.14.3.9 Öffnen von Bibliotheken aus der Resultview – 08.06.2016

Aufgrund von Verzögerungen im Zeitplan wurde aus der Reservewoche eine 5. Iteration gemacht. Diese Iteration war für die Implementation nötig, da das Öffnen von Bibliotheken Schwierigkeiten bereitete. In dieser wurde dann das Öffnen von Bibliotheken komplett implementiert.

8.14.3.10 Dokumentation – 15.06.2016

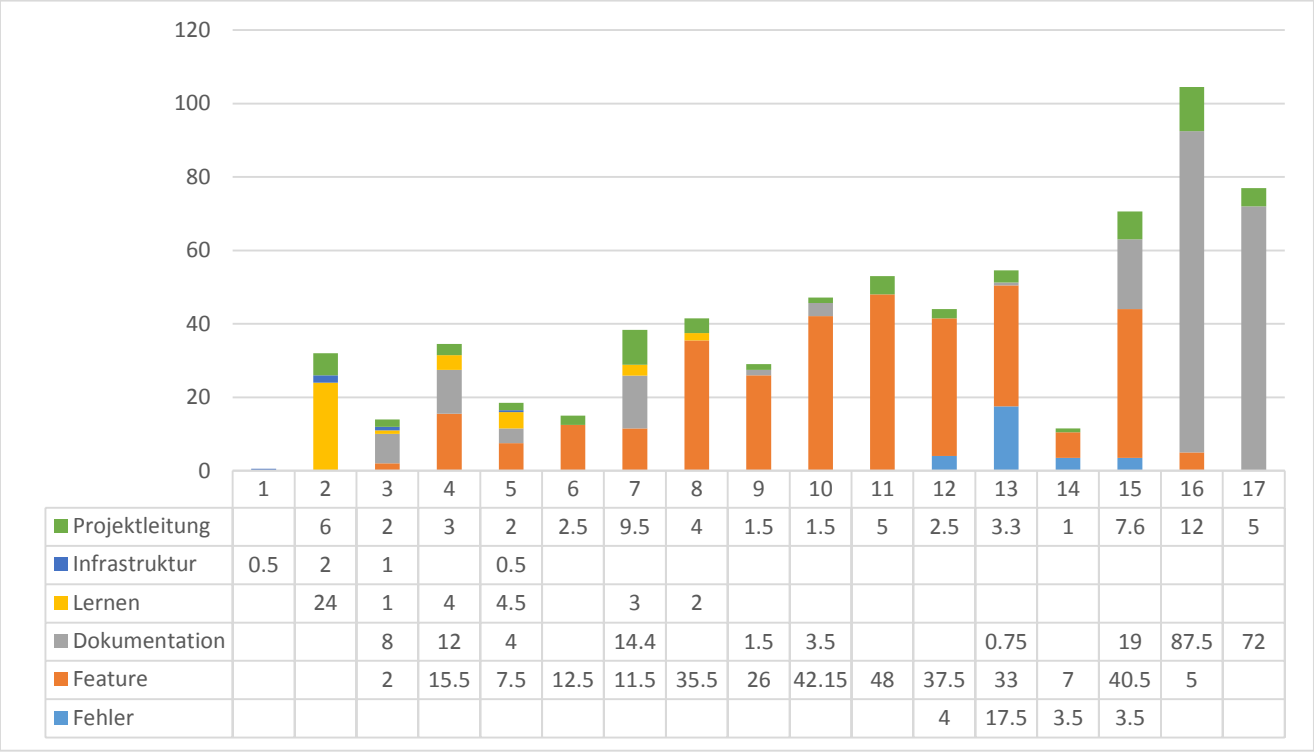
Die ganze Dokumentation soll bis zu diesem Zeitpunkt geschrieben sein. Dies beinhaltet den technischen Bericht, Management Summary, Abstract und die persönlichen Berichte. Das Projekt soll zur Abgabe bereit sein.

8.14.4 Qualitätssicherung

Um die Qualität der Arbeit sicherzustellen, wurden Dokumente und Code Reviews gemacht.

Der Code Review wurde am 20.05.2016 von Mirko Stocker durchgeführt. Die Kommentare können hier angesehen werden: https://github.com/flomerz/scala-ide-scaps/pull/2/files. Alle Punkte wurden angeschaut und behandelt.

8.14.5 Zeiterfassung



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Projektleitung		6	2	3	2	2.5	9.5	4	1.5	1.5	5	2.5	3.3	1	7.6	12	5
Infrastruktur	0.5	2	1		0.5												
Lernen		24	1	4	4.5		3	2									
Dokumentation			8	12	4		14.4		1.5	3.5			0.75		19	87.5	72
Feature			2	15.5	7.5	12.5	11.5	35.5	26	42.15	48	37.5	33	7	40.5	5	
Fehler												4	17.5	3.5	3.5		

8.14.6 Code-Statistiken

Hier sind einige Auszüge der Git-Statistiken:

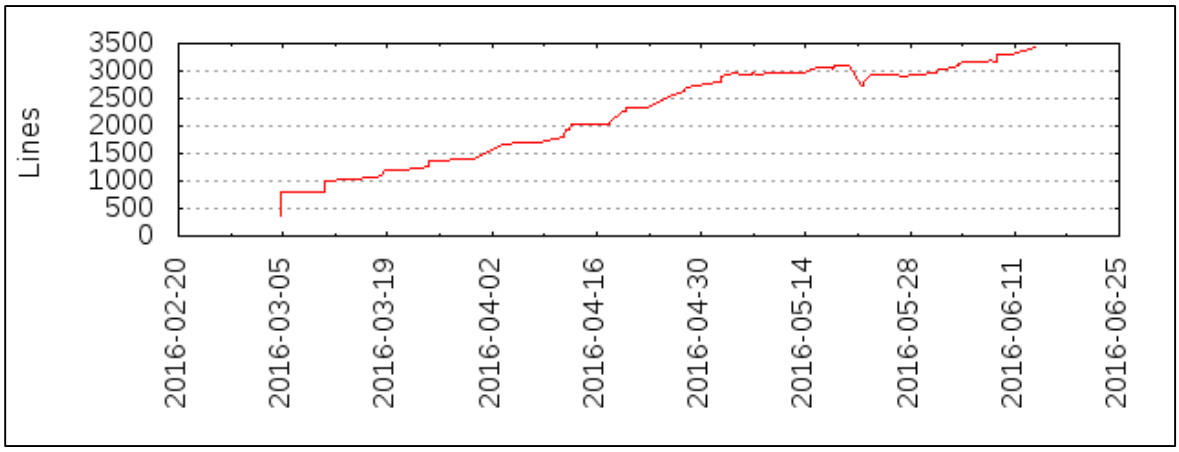


Abbildung 24: Anzahl Code Zeilen

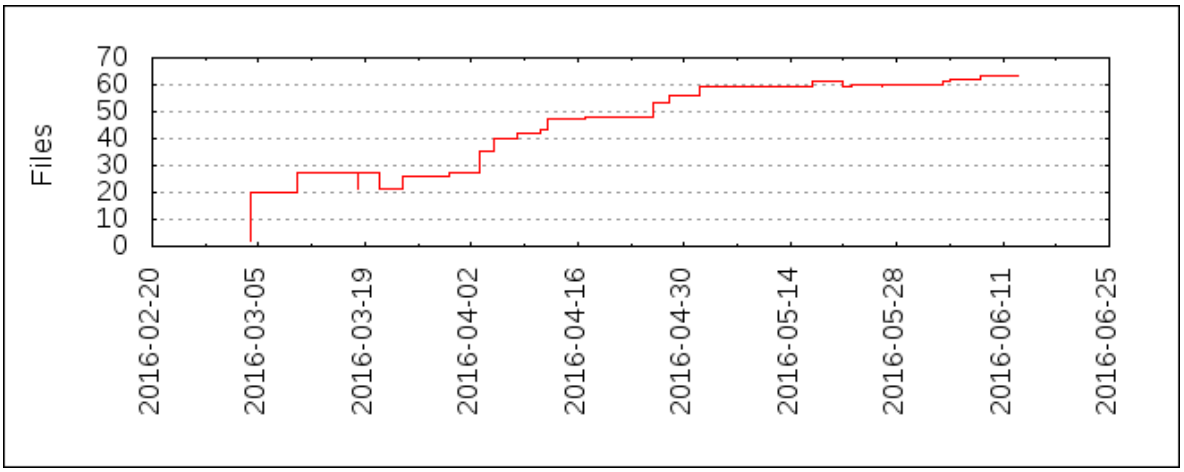


Abbildung 25: Anzahl Dateien

Extension	Files (%)	Lines (%)	Lines/file
	15 (24.19%)	931 (27.14%)	62
MF	2 (3.23%)	94 (2.74%)	47
conf	1 (1.61%)	28 (0.82%)	28
gif	1 (1.61%)	0 (0.00%)	0
md	1 (1.61%)	2 (0.06%)	2
properties	3 (4.84%)	60 (1.75%)	20
scala	31 (50.00%)	1884 (54.93%)	60
xml	8 (12.90%)	426 (12.42%)	53

Abbildung 26: Informationen zu den Dateien

9 Nutzungsrechte

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Vereinbarung

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Bachelorarbeit *Scala Search IDE-Integration* von Royce Manavalan und Florian Merz unter der Betreuung von Mirko Stocker geregelt.

2. Urheberrecht

Die Urheberrechte stehen der Studentin / dem Student zu.

3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von der Studentin / dem Student wie von der HSR nach Abschluss der Arbeit verwendet und weiter entwickelt werden

Rapperswil, den 16.3.16 Florian Merz
Die Studentin/der Student

Die Studentin/der Student

Rapperswil, den 17.3.16 M. Stu
Der Betreuer

Rapperswil, den 23.3.16
Der Studiengangleiter / die Studiengangleiterin

1

10 Quellen

10.1 Eclipse

- Eclipse 4 Plugin-in Development by Example Dr Alex Blewitt
- Eclipse 4, Rich Clients mit dem Eclipse SDK 42 Marc Teufel, Dr. Jonas Helming
- https://docs.oracle.com/javase/8/docs/api/java/util/zip/ZipEntry.html
- https://wiki.eclipse.org/FAQ_How_do_I_open_an_editor_on_a_file_outside_the_workspace%3F
- http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Freference%2Fapi%2Forg%2Feclipse%2Fcore%2Fresources%2FIResource.html&anchor=getLocation%28%29
- http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2FresInt_filesystem.htm
- http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Freference%2Fapi%2Forg%2Feclipse%2Fcore%2Fresources%2FIWorkspace.html
- http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Freference%2Fapi%2Forg%2Feclipse%2Fui%2Fide%2FIDE.html
- https://wiki.eclipse.org/FAQ_How_do_I_display_search_results%3F
- https://wiki.eclipse.org/FAQ_How_do_I_write_a_Search_dialog%3F
- http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fsearch_page.htm
- http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fsearch_result.htm
- https://wiki.eclipse.org/FAQ_How_do_I_open_an_editor_programmatically%3F
- https://wiki.eclipse.org/FAQ_How_do_I_implement_Quick_Fixes_for_my_own_language%3F
- http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2FresAdv_markers.htm&anchor=resAdv_markers_extending
- http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Freference%2Fextension-points%2Forg_eclipse_ui_workingSets.html
- http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.jdt.doc.isv%2Fguide%2Fjdt_api_contributing_a_quickfix.htm
- http://wiki.eclipse.org/FAQ_How_do_I_write_a_Search_dialog%3F
- http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.jdt.doc.isv%2Fguide%2Fjdt_api_search.htm
- http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.jdt.doc.isv%2Fguide%2Fjdt_api_codeassist.htm&cp=3_0_0_6
- http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.jdt.doc.isv%2Freference%2Fapi%2Forg%2Feclipse%2Fjdt%2Fcore%2Fsearch%2FSearchMatch.html
- http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Ffirstplugin.htm

- http://www.wideskills.com/eclipse-plugin-tutorial/introduction-to-eclipse-plugin-development
- http://www.vogella.com/tutorials/EclipsePlugIn/article.html

10.2 Scala

- http://www.tutorialspoint.com/scala/scala_options.htm
- http://joelabrahamsson.com/learning-scala-part-four-classes-and-constructors/
- http://www.tutorialspoint.com/scala/scala_pattern_matching.htm

10.3 Scaps

- http://about.scala-search.org/thesis.pdf
- https://github.com/scala-search/scaps