Requirements Engineering Environment (RENE)

Praktikumsbericht im Fach Informatik
vorgelegt von

Martin Noack
Matrikelnummer 09-222-232

Michael Studer
Matrikelnummer 09-714-551

Angefertigt am
Institut für Informatik
der Universität Zürich
Prof. Dr. M. Böhlen

Betreuer:
Claudio Jossen, Credit Suisse
Dietrich Christopeit, Credit Suisse

Abgabe der Arbeit: 10. Juli 2012

Inhaltsverzeichnis

1. Einleitung

Die Studienordnung in Informatik an der Universität Zürich sieht für jeden Studenten der Fachrichtung Softwaresysteme verpflichtend die Durchführung eines externen Softwareprojekts vor. Dieser Bericht beschreibt unser Projekt bei der Firma Credit Suisse, welches wir über einen Zeitraum von 9 Wochen durchführen konnten. Beginnend am 16. April bis einschliesslich 22. Juni 2012 hatten wir die Gelegenheit unter intensiver Betreuung einen lauffähigen Prototypen des „Requirements Engineering Environment“ zu konzipieren und implementieren. Hierbei erhielten wir einen äusserst interessanten Einblick in die Arbeitsstrukturen der Credit Suisse sowie dem Software Entwicklungsprozess in der Praxis im Allgemeinen.
In diesem Bericht werden wir auf die genaue Aufgabenstellung, unser Vorgehen bei der Lösungserarbeitung sowie gemachte Erfahrungen und gesammelte Eindrücke detailliert eingehen.
Da die Applikation allerdings mit internen Daten der Credit Suisse arbeitet, können wir an dieser Stelle gemäss dem non-disclosure agreement nur exemplarische Platzhalter in den Screenshots verwenden und nur die neutralen Kernelemente der Anwendung aufzeigen. Vor Beginn des Projekts wurden wir daher explizit auf diese Einschränkungen aufmerksam gemacht und unterschrieben den detaillierten Geheimhaltungsvertrag.

2. Aufgabenstellung

Ziel des Projekts war die Entwicklung eines lauffähigen Prototyps der Requirement Engineering Environment (im weiteren RENE genannt) nach den Standards der Credit Suisse. Im Detail dient RENE als Eingabemaske zur Dokumentation sogenannter Mappings im Umfeld der Credit Suisse. In Datawarehouse-Umgebungen ist es oft erforderlich, Daten von verschiedenen Quellen/Quelltabellen zu extrahieren und in einer Zieltabelle bereitzustellen. Dies geschieht derart, dass Regelsätze angelegt werden, nach denen

- gesamte Quell-Tabellen in Ziel-Tabellen überführt werden
- einzelne Attribute von Quell-Tabellen Attributen den Ziel-Tabellen zugewiesen werden

Mappings, wie hierbei die Regelsätze dieser Abbildungen genannt werden, enthalten insbesondere auch Funktionen wie arithmetische Berechnungen, Umbenennungen, Aggregationen und andere Bedingungen.
Die Eingabemaske sollte dabei so konzipiert sein, dass sie dem Benutzer intuitiv verständlich ist und alle bekannten Anforderungen an die Beschreibung von Mappings erfüllt. Insbesondere sollten dabei alle Möglichkeiten der Datenbanktechnik aber auch deren Sinnhaftigkeit in diesem Anwendungsfall berücksichtigt werden.
Nach kurzer Zeit wurde uns klar, dass noch keine internen Standards zur Beschreibung von Mappings existieren. Daraus ergab sich notwendigerweise eine weitere Aufgabe: die Konzipierung einer einheitlichen Syntax zur universalen Darstellung von Mappings. Zu diesem Zweck haben wir eng mit dem gesamten Team und insbesondere dem Datenmodellierer zusammengearbeitet, um sämtliche eventuell auftauchenden Requirements akkurat abzudecken.
Aus dieser Syntax sollte nun direkt eine Grammatik erstellt werden, die wiederum zum Überprüfen der eingegebenen Mappings dienen kann. Diese Prüfung war weiterhin im Prototyp als syntaktischer Check mit rudimentärer Fehlerausgabe zu implementieren.
Als Framework war csJSF zu nutzen, die firmeneigene Variante von JSF. Dies sollte uns später vor einige Herausforderungen stellen, insbesondere auf den komplexen Datenbankzugriff mussten wir daher aus Zeitgründen größtenteils verzichten.
Zu den oben genannten Standards gehörten hauptsächlich eine ausführliche Dokumentation in Form von Screen Requirements, Screen Requirement Specifications, Business Rule Specifications, Use Case-Beschreibungen, UML-Klassendiagrammen, einem detaillierten Physical Design, JavaDoc und ein User-Manual.

- Screen Requirements bestehen aus einem groben Entwurf der Anwendungsoberfläche. Nachdem die Anforderungen erhoben wurden, kann damit exemplarisch der Arbeitsablauf in der Anwendung gezeigt werden. In unserem Falle erstellten wir eine Reihe von Folien in Powerpoint, die wie Screenshots der Anwendung aufgebaut waren und alle geplanten Funktionalitäten vorführten.
- Screen Requirement Specifications dienen der ausführlichen Beschreibung aller in den Screen Requirements vorgestellten Funktionen. Dies geschieht tabellarisch für jedes einzelne Screen Requirement. Insbesondere die genauen Reaktionen auf die Verwendung von Buttons sowie Verhaltensregeln für Statustexte sind zu beschreiben.

- Business Rules beinhalten alle formalen Regeln, die für die Anwendung aufgestellt wurden. Dies können beispielweise Regelungen zur Autovervollständigung von Worten sein, oder wie in unserem Falle die verwendete Syntax für Mappings.
- Use Cases wurden in bekannter Art und Weise erstellt, um alle Anwendungsfälle genau zu beschreiben.
- UML-Klassendiagramme beschreiben ebenfalls wie üblich die verwendeten Klassen und deren Beziehungen untereinander.
- Das Physical Design erläutert jede einzelne Klasse im Detail, dabei werden die wichtigsten Methoden und Attribute in Klartext oder Pseudocode erläutert. Es wird ebenfalls tabellarisch dargestellt und ergänzt das JavaDoc um technische Erläuterungen, welche die Klassen in den Gesamtkontext setzten.
- Das User-Manual wurde in das interne Wiki integriert und soll dem Benutzer die Bedienung der Anwendung verdeutlichen. Anhand von Screenshots und detaillierten Erklärungen wird exemplarisch der komplette Workflow dargestellt.

Wie klar zu erkennen ist, wird die Dokumentation bei Credit Suisse ausserordentlich sorgfältig und umfangreich gestaltet. Dies liegt unter anderem daran, dass die eigentliche Programmierung häufig an ein eigenes Team ausgelagert wird. Die Mitglieder dieses Teams müssen also in der Lage sein, die beschriebene Software anhand der Dokumentation vollständig zu verstehen und zu implementieren.
Zusammenfassend bestand unser Projekt also aus der vollständigen Entwicklung eines lauffähigen RENE-Prototyps, beginnend bei der Anforderungserhebung bis hin zur kompletten Implementierung und Dokumentation.

3. Technische Umsetzung

In diesem Kapitel des Berichtes soll nun näher auf die technische Umsetzung eingegangen werden. Während des neunwöchigen Praktikums entstand nicht nur ein experimenteller Prototyp, wie er explizit in der Aufgabenstellung gefordert wurde, sondern auch eine eigenständige allgemeine Grammatik für Mappings. Diese Grammatik wurde von Grund auf entwickelt, da es sich während der Arbeit am Prototyp zeigte, dass bislang noch keine einheitliche Syntax für Mappings existiert. Solch ein Standard birgt jedoch offensichtliche Vorteile. So erlaubt eine zugrundeliegende Grammatik unter Anderem neu verfasste oder geänderte Mappings anhand grammatikalischer Regeln (automatisch) auf syntaktische Korrektheit zu überprüfen.

3.1 Grammatik

Wie bereits in der Aufgabenstellung beschrieben ergab sich relativ schnell die Notwendigkeit, Mappings durch einen einheitlichen Standard auszudrücken. Obwohl dieser Teil ursprünglich nicht direkt mit der Entwicklung der Anwendung verknüpft war, entwickelte sich die Grammatik schnell zu einem zentralen Element, welches auch ausserhalb der RENE Applikation von Bedeutung ist.
Angestossen wurde der Entwicklungsprozess durch die Spezifikation einer allgemeinen Darstellungsform von Mappings. Bisher wurde die Dokumentation formlos in Excel-Tabellen oder Word-Dokumenten festgehalten. Dabei fehlte eine einheitliche Darstellung ebenso wie eine verbindliche Syntax.
Es existierte allerdings bereits ein Datenmodell, das den Aufbau von Mappings allgemein beschreibt. Dieses Modell diente uns als grober Ausgangspunkt. Allerdings wurde während dem Entwicklungsprozess klar, dass dieses Datenmodell bei weitem nicht alle Anforderungen an die Spezifikationen erfüllen konnte.
Zu Beginn erstellten wir im Team eine rudimentäre Syntax nach den Vorgaben unserer Betreuer und dem Datenmodellierer. Da die Beschreibung der Mappings letztendlich jedoch nicht nur als Vorlage für Spezialisten dienen sollte, sondern auch als intuitiv verständliche Dokumentation, wurde bewusst entschieden, dass man sich vom SQL-Stil lösen sollte. So kann zwar beispielsweise ein Join eine Abbildung akkurat beschreiben, ist aber nicht verständlich für Personen, die kein Vorwissen über Datenbanken besitzen. Somit wurden mehrheitlich intuitiv verständliche Schlüsselwörter verwendet, die ein Mapping sinngemäß wiedergeben. Beispielsweise beginnt ein Mapping mit „WHEN POPULATING <target> FROM <source> […]“, wobei <target> für die zu füllende Tabelle steht und <source> für die Tabelle, aus der hauptsächlich Daten entnommen werden. Dieser Ausdruck beschreibt in keinster Weise die vollständige Abbildung aus Datenbanksicht, aber erlaubt es einer Mehrheit von Personen sofort konzeptionell zu verstehen, auf welche Tabelle die Abbildung abzielt und woher die Daten hauptsächlich stammen. Abbildung 3.1 zeigt einen Ausschnitt aus der Grammatik, wobei <mapping> als Startpunkt dient.

```
<mapping> ::= <population> [ <filter> ] [ <navigation> ]

<population> ::= “WHEN POPULATING“ <tEntity> “FROM“ <dEntity> <population-
      Clause>

<populationClause> :: = “POPULATE“ <tAttribute> <withClause> |
      “POPULATE“ <tAttribute> <withClause> <eol> <populationClause>

<withClause> ::= “WITH” <expression> “IF” <condition> | “WITH” <expression> |
      “WITH” <expression> “IF” <condition> <eol> <withClause>

<expression> ::= <word> | <computation>

<computation> ::= <attribute> | <Value> | <aggregation> |
      [“(”] <computation> <operator> <computation> [“)”]
```

Abbildung 3.1: Ausschnitt aus der Grammatik

Anhand dieser Syntax konnten wir nun eine Grammatik entwickeln und in der Backus-Naur-Form darstellen. Dieses diente sowohl als Vorlage für einen Parser, als auch zur Überprüfung des zugrunde liegenden Modells. Im Gespräch mit dem Datenmodellierer konnten wir so Schwachstellen im Datenmodell aufdecken und gleichzeitig unser Verständnis der Anforderungen formal validieren.
Mit Hilfe des Parsergenerators ANTLR[1] war es nun verhältnismässig einfach einen Parser zu gestalten, der die eingegebenen Mappings auf korrekte Syntax prüfen kann. Dabei waren lediglich einige technische Änderungen an der Grammatik nötig, die sich aus der Funktionsweise von ANTLR ergaben. Insbesondere mussten alle rekursiven Regeln durch äquivalente reguläre Ausdrücke ersetzt werden. Die generierten Klassen liessen sich problemlos in das Projekt einbinden.
Unsere Arbeit in Hinsicht auf die Syntax war mit der Entwicklung des Syntaxprüfers abgeschlossen, jedoch wären noch einige weitere Entwicklungen denkbar. So kann momentan nicht auf Sinnhaftigkeit der Mappings geprüft werden, eine Abbildung von einer Tabelle auf sich selbst ist in unserer Implementierung ebenso valide wie die Verwendung von nicht-existenten Tabellen.

3.2 Web-Applikation „RENE“

Neben der Grammatik, welche im vorangehenden Kapitel näher beschrieben wurde, entwickelten wir während des Praktikums einen experimentellen Prototyp in Form einer Web-Applikation, welche den Richtlinien der Credit Suisse genügt. Dabei baut diese Applikation neben standardisierten Beschreibungs-, Skript- und Programmiersprachen (namentlich sind dies HTML, JavaScript und Java) auch auf bestehenden Frameworks auf, welche in der Credit Suisse fortlaufend entwickelt und gewartet werden. Diese Frameworks bieten dem Entwickler in erster Linie die Funktionalität für die Erstellung von graphischen Benutzeroberflächen, ganz dem Corporate Identity der Credit Suisse entsprechend. Zusätzlich kann mit Hilfe dieser Frameworks die Authentifizierung eines Benutzers geregelt werden. Jegli-

[1] http://www.antlr.org/

che andere Funktionalität hingegen, wie beispielsweise logische Programmabläufe, Zugriffe auf Datenbanken oder die Speicherung persistenter Daten, müssen vom Entwickler selber erstellt werden. Dies geschieht normalerweise auf der Basis von so genannter Beans. Dies sind typischerweise Klassen in Java, die eine abgegrenzte Menge an Funktionalität einer Applikation kapseln und dieser dann zur Verfügung stellen.
Es liegt auf der Hand, dass in einer international tätigen Schweizer Bank, wie es die Credit Suisse darstellt, Daten innerhalb verschiedenster Applikationen neu erstellt oder bearbeitet werden. Diese Daten werden wiederum in unzähligen Datenbanken gespeichert. Es ergibt daher wenig Sinn, sämtliche Mappings oder gar Entitäten, welche in einem Mapping benutzt werden, dem Benutzer in einer überdimensionierten Liste zu präsentieren. Einerseits wäre ein solches Vorgehen in keinster Weise benutzerfreundlich, andererseits würde dies eine unnötige Last für sämtliche Datenbanksysteme bedeuten. Daher entschlossen wir uns, unseren experimentellen Prototypen dahingehend zu gestalten, dass zuerst eine Wahl auf sämtlichen hierarchischen Ebenen getroffen werden muss, bevor vorhandene Mappings oder verfügbare Entitäten aufgelistet werden. Die folgende Tabelle gibt einen Überblick über sämtliche hierarchischen Ebenen.

Stufe	Hierarchische Ebene
1. Stufe (oberste Stufe)	Domänen
2. Stufe	Applikationen
3. Stufe	Informationsgruppen
4. Stufe	Informationsuntergruppen
5. Stufe (unterste Stufe)	Entitäten

Tabelle 3.1: Übersicht über sämtliche hierarchischen Ebenen

An erster Stelle steht die Wahl einer Domäne. Alle verfügbaren Domänen werden nach dem Start der Web-Applikation übersichtlich in Form einer Illustration dem Benutzer zur Wahl angeboten. Nach einer getätigten Wahl werden in einem nächsten Schritt sämtliche Applikationen, die innerhalb der gewählten Domäne zur Verfügung stehen, in einer geordneten Liste dargestellt. Dies wiederholt sich schliesslich für die Informationsgruppen, Informationsuntergruppen und Entitäten einer gewählten Applikation. Die nachfolgende Abbildung zeigt nun die Liste der Informationsgruppen, Informationsuntergruppen und Entitäten.

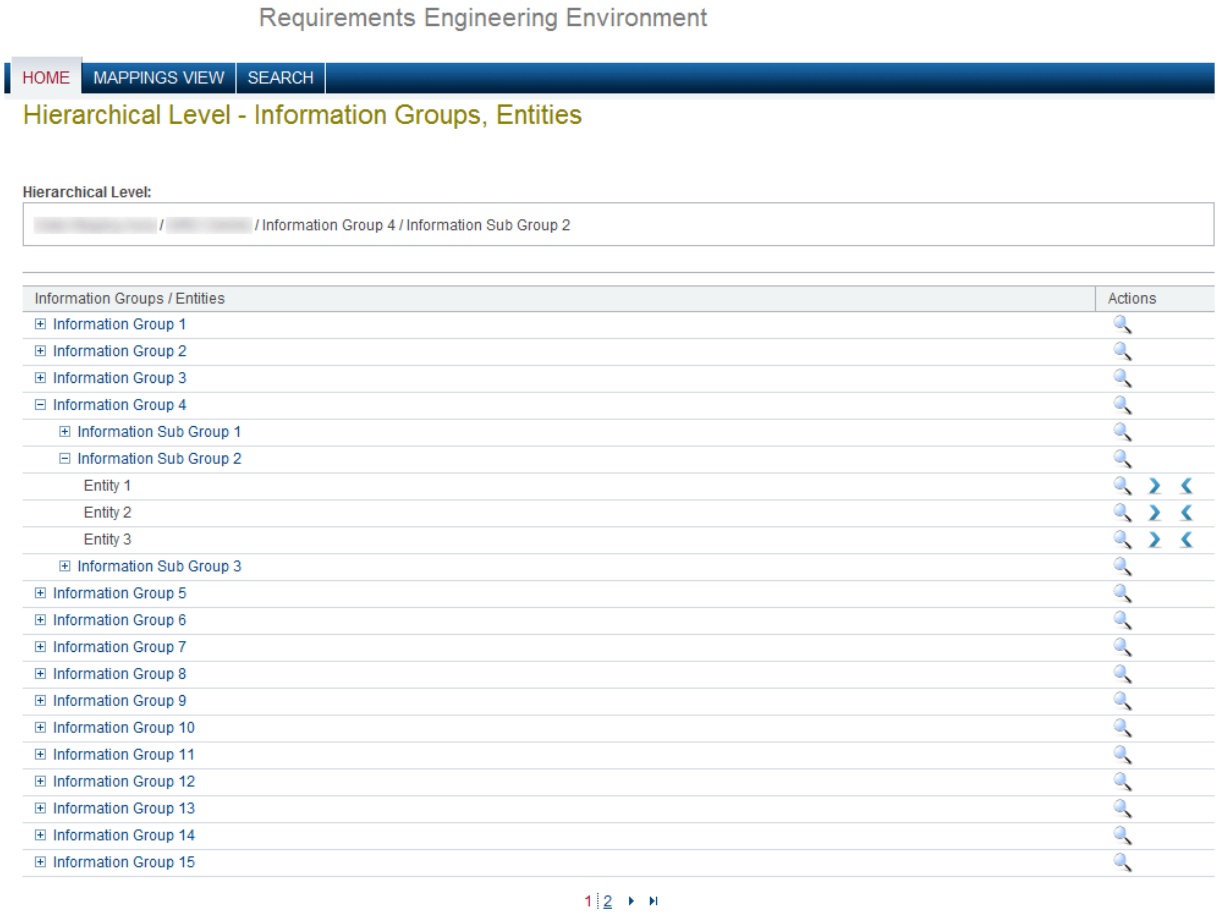

Abbildung 3.2: Informationsgruppen, Informationsuntergruppen und Entitäten einer bestimmten Applikation

Zu jedem Zeitpunkt, also während der Wahl einer Domäne, einer Applikation, einer Informationsgruppe, einer Informationsuntergruppe oder einer Entität, ist unterhalb der Navigationsleiste (erkennbar an den Navigationspunkten „Home“, „Mappings View“ und „Search“) und der Titelzeile die aktuelle hierarchische Ebene sichtbar, auf welcher sich der Benutzer gerade bewegt. Mit einem einfachen Klick auf eine Informationsgruppe werden die Informationsuntergruppen der entsprechenden Informationsgruppe angezeigt. In der Abbildung oben ist es die Informationsgruppe „Information Group 4“. In gleicher Weise bewirkt ein Klick auf eine Informationsuntergruppe (in der Abbildung 3.1 wurde die Informationsuntergruppe „Information Sub Group 2“ gewählt) eine Auflistung sämtlicher Entitäten der gewählten Informationsuntergruppe. An diesem Punkt erwähnenswert sind nun die drei Symbole Lupe, rechtsweisender Pfeil sowie linksweisender Pfeil, wobei nur Entitäten die beiden Pfeile besitzen. Wie ein Benutzer intuitiv erwarten würde, bedeutet ein Klick auf die Lupe das Öffnen eines kleinen Fensters mit zusätzlichen Informationen, wie beispielsweise dem Namen, dem Code oder dem Kommentar der entsprechenden Informationsgruppe, Informationsuntergruppe oder Entität. Viel interessanter ist jedoch die Funktion der blauen Pfeile. Dabei symbolisiert ein Klick auf den rechtsweisenden Pfeil, dass die entsprechende Entität als so genannte „Driving Entity“ betrachtet werden soll, während analog dazu der linksweisende Pfeil die entsprechende Entität als „Target Entity“ auffasst. In diesem Kontext bedeutet „Driving Entity“, dass die entsprechende Entität als

Hauptquelle für das Mapping dient, und „Target Entity“, dass die Entität als Hauptziel dient. Klickt der Benutzer nun auf einen der beiden Pfeile, wird auf einer nächsten Seite sämtliche Mappings übersichtlich aufgelistet, welche die gewählte Entität entweder als „Driving Entity“ (Hauptquelle) oder als „Target Entity“ (Hauptziel) verwendet, je nachdem, welcher Pfeil gewählt wurde. Die untenstehende Abbildung zeigt diese Mappings-Liste.

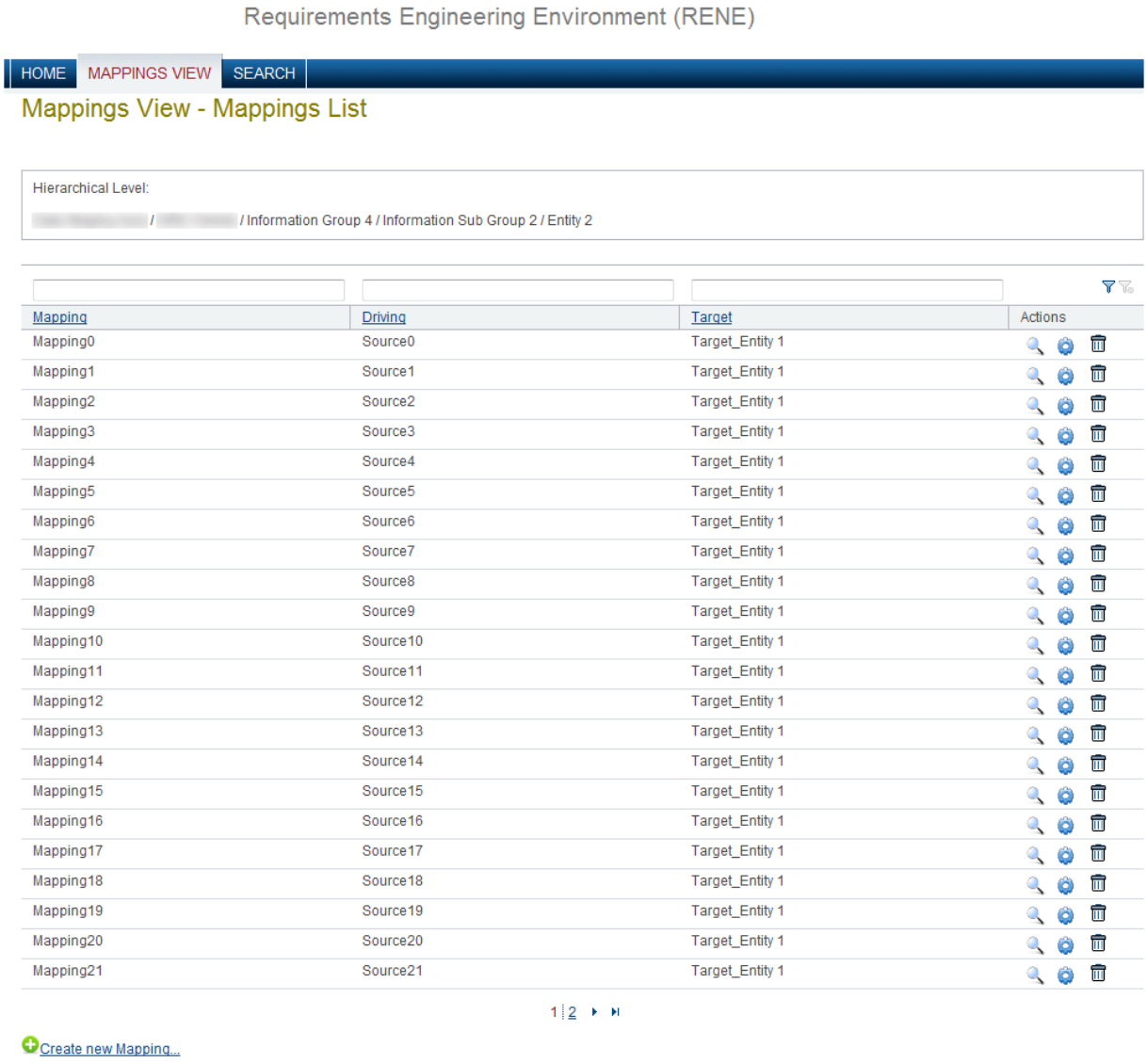

Abbildung 3.3: Sämtliche Mappings mit dem Hauptziel „Target_Entity 1“ übersichtlich aufgelistet

Wie es bereits auf der Seite zur Wahl der Applikation, der Informationsgruppe, der Informationsuntergruppe und der Entität der Fall war, befindet sich auch auf dieser Seite unter der Navigationsleiste ein Feld mit der aktuell gültigen hierarchischen Ebene. Gleich unter diesem Feld sind drei leere Textfelder positioniert. Diese Textfelder bilden zusammen einen Filter, der mit den zwei Trichter-Symbolen rechts vom dritten Textfeld ein- beziehungsweise ausgeschaltet werden kann. Gibt der Benutzer nun einen vollständigen Namen (oder auch nur Teile davon) eines Mappings oder einer Entität in das entsprechende Textfeld ein und aktiviert den Filter mit dem Trichter-Symbol, wird die Liste automatisch aktualisiert und es erscheinen nur noch diejenigen Mappings in der Liste, welche der Eingabe

im Filter entsprechen. Unabhängig von diesem Filter besitzt jedes aufgelistete Mapping wiederum drei Symbole, deren Funktionen intuitiv erkennbar sein sollten. Das Lupen-Symbol zeigt, wie bereits bekannt, ein zusätzliches Fenster mit Informationen. In dieser Liste ist es der Name sowie die Beschreibung des Mappings. Neu hingegen ist das Zahnrad-Symbol. Es öffnet das entsprechende Mapping im eigens entwickelten Editor zur Bearbeitung. Schliesslich löscht das Papierkorb-Symbol das ausgewählte Mapping. Letzteres wird natürlich nur durchgeführt, wenn der Benutzer eine Sicherheitsabfrage positiv beantwortet, um versehentliches Löschen von Mappings zu vermeiden. Ganz zuunterst auf der Seite befindet sich die Schaltfläche „Create new Mapping...“, mit welcher der Benutzer ein neues Mapping erstellen kann. In diesem Falle wird wiederum die Editor-Seite geöffnet, jedoch mit leerem Inhalt. Die folgende Abbildung zeigt die Editor-Seite mit geöffnetem Mapping „Mapping12“.

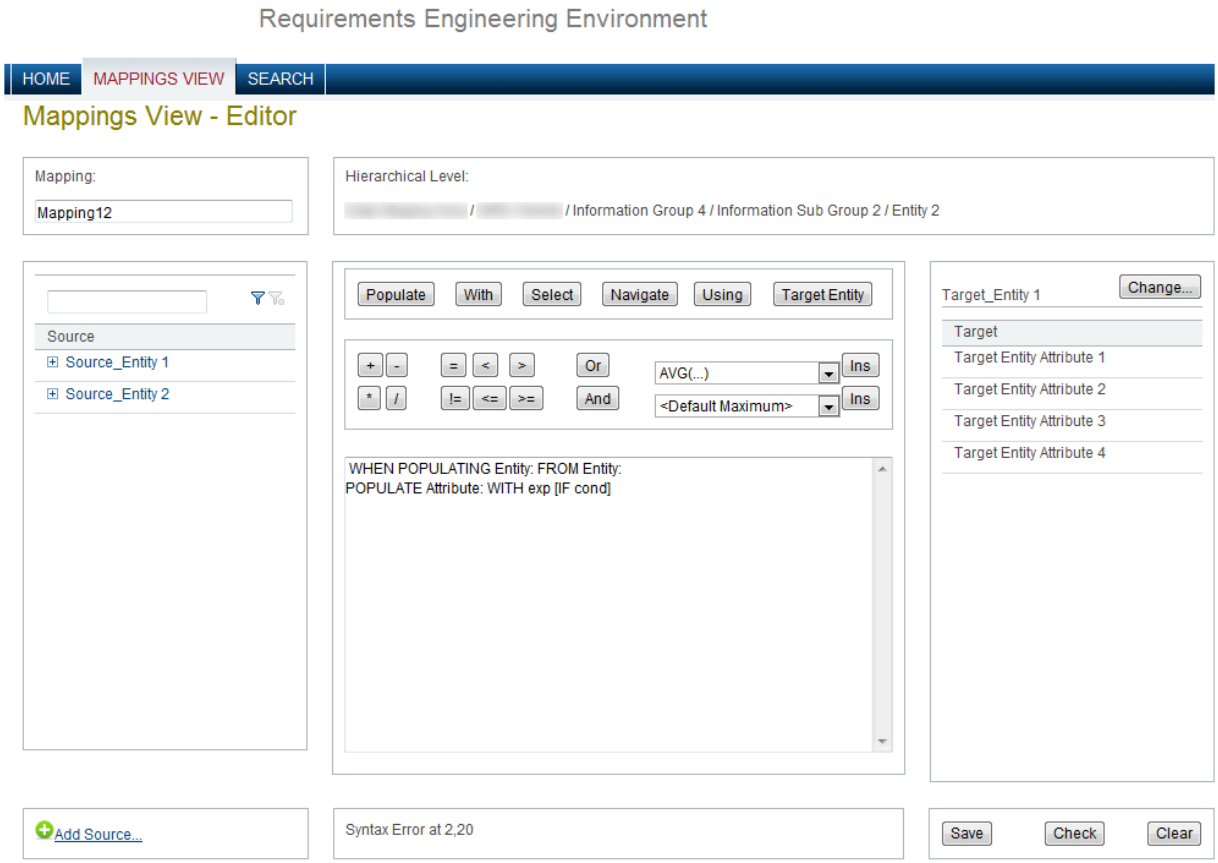

Abbildung 3.4: Die Mapping Editor-Seite mit geöffnetem Mapping „Mapping12“

Die Editor-Seite für Mappings, das eigentliche Kernstück der „RENE“ Web-Applikation, ist in drei Teile gegliedert. Der obere Teil, gleich unterhalb der bekannten Navigationsleiste positioniert, zeigt links in einem editierbaren Textfeld den Namen des aktuell dargestellten Mappings, während sich rechts davon wiederum das Feld mit der hierarchischen Ebene befindet. In der Mitte der Editor-Seite ist der Hauptteil zu sehen, der mit Abstand am meisten Platz einnimmt. Von links nach rechts betrachtet findet der Benutzer in diesem die „Source“-Liste mit der „Driving Entity“ sowie den „Source Entities“ (also die Hauptquelle und zusätzliche Quellen), das eigentliche Editor-Feld (weiter unten mehr dazu) und am

rechten Rand die „Target Entity“ (das Hauptziel, welches mit der Schaltfläche „Change...“ geändert werden kann) sowie sämtliche Attribute der gewählten „Target Entity“. Der untere Abschnitt der Editor-Seite ist schliesslich dreigeteilt. Der linke Teil beinhaltet den Schaltknopf „Add Source...“, mit welchem eine zusätzliche Quelle zu der Quellenliste „Source“ hinzugefügt werden kann, der mittlere Teil zeigt Meldungen zu der syntaktischen Korrektheit des Mappings an, während der rechte Teile die drei selbsterklärenden Schaltflächen „Save“, „Check“ und „Clear“ beherbergt. (Nähere Informationen zu der syntaktischen Korrektheit von Mappings findet man im Kapitel „3.1 Grammatik“.)
Bei der Erstellung beziehungsweise bei der Bearbeitung eines Mappings konzentriert sich der Benutzer die meiste Zeit über auf den zentralen Teil der Editor-Seite. Namentlich sind dies die kontextuell passend angeordneten und gruppierten Schaltflächen sowie den eigentliche Editor-Bereich. Um nun dem Benutzer eine wertvolle Hilfestellung bei seiner Arbeit an den Mappings zu geben, entschlossen wir uns, ihn nicht mit einem grossen Textfeld alleine zu lassen. Stattdessen entwickelten wir eine Benutzeroberfläche ähnlich eines Taschenrechners, auf welcher der Benutzer mit einem Blick sieht, welche Möglichkeiten ihm gerade zur Verfügung stehen, und mit welcher er sich das Mapping wie in einem Baukasten „zusammenbauen“ kann. Dies funktioniert deshalb, da mit jedem einzelnen Klick auf eine Schaltfläche eine entsprechende Anweisung (ein Statement) in das Mapping eingefügt wird. Dabei kann eine solche Anweisung entweder aus einem ganzen Anweisungsblock, aus einer einfachen Anweisung, aus einem mathematischen Operator, aus einem Vergleichsoperator, aus einem logischen Operator, aus einer Aggregation, oder aus einem so genannten Makro bestehen. Folgende Tabelle gibt einen Überblick über die einzelnen Anweisungen, die im Editor verfügbar sind. (Nähere Informationen zu den einzelnen Anweisungen in Mappings findet man auch im Kapitel „3.1 Grammatik“.)

Anweisungsart	Schaltfläche im Editor
Anweisungsblock	„Populate“, „Select“ oder „Navigate“
Einfache Anweisung	„With“, „Using“ oder „Target Entity“
Mathematischer Operator	+, –, * oder /
Vergleichsoperator	=, <, >, !=, <= oder >=
Logischer Operator	„And“ oder „Or“
Aggregation	„AVG(...)“, „MAX(...)“, „MIN(...)“ oder „SUM(...)“
Makro	„<Default Maximum>“, „<Default Minimum>“, „<Not Available>“, „<Not Known>“ oder „<Not Used>“

Tabelle 3.2: Verschiedene Anweisungsarten und ihre Zugehörigkeit im Mapping-Editor

Natürlich müssen eventuelle Platzhalter, welche von der Web-Applikation nicht automatisch ausgefüllt werden können, manuell nachgeführt werden. Auf jeden Fall steht es dem Benutzer offen, gänzlich auf die Verwendung von Anweisungs-Schaltflächen zu verzichten. Dabei kann der Benutzer über die Tastatur ein komplettes Mapping im Textfeld eingeben. Schliesslich hat der Benutzer nach dem Erstellen oder der Bearbeitung eines Mappings die Möglichkeit, mit der Schaltfläche „Clear“ das Mapping zu verwerfen und neu zu beginnen, mit „Check“ das Mapping auf syntaktische Fehler überprüfen zu lassen, oder mit „Save“ das neue oder geänderte Mapping zu speichern.
Um die gesamte Funktionalität der Web-Applikation „RENE“ abzurunden, bauten wir zusätzlich eine übersichtliche Suchfunktion ein, welche zu jedem Zeitpunkt über den Navigationspunkt „Search“ zu erreichen ist. Die Suche erlaubt es, nach einem Mapping-Namen,

nach einer verwendeten „Driving Entity“, oder nach einer verwendeten „Target Entity“ zu suchen. Die untenstehende Abbildung zeigt die Such-Seite mit Suchresultaten.

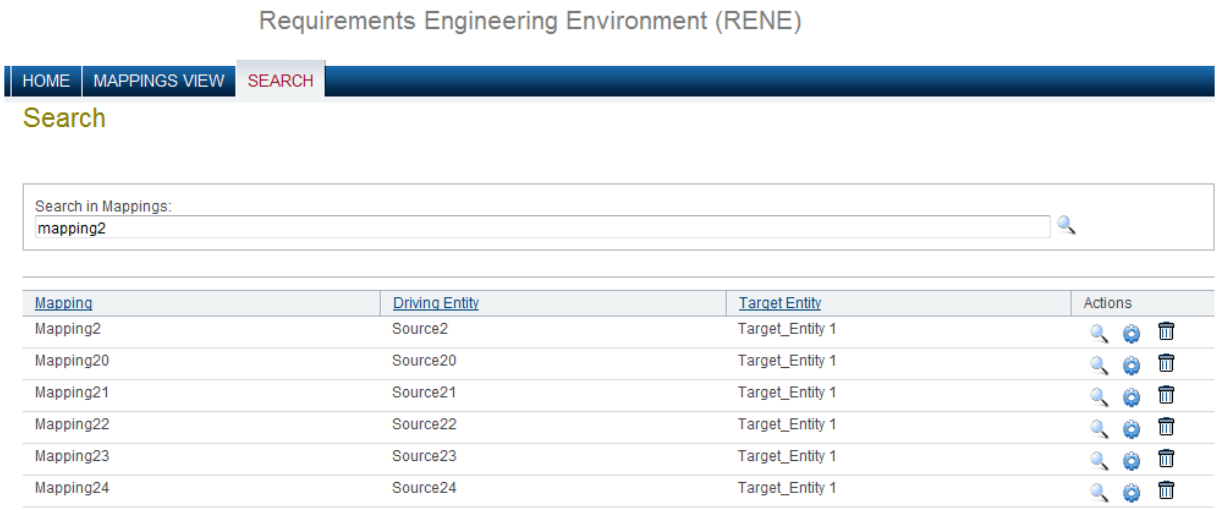

Abbildung 3.5: Das Suchresultat einer Suche nach Mappings

Wie allgemein von Suchmaschinen bekannt befindet sich oberhalb einer möglichen Resultatsliste ein Textfeld für den eigentlichen Suchbegriff. Mit einem Klick auf das Lupen-Symbol rechts dieses Textfeldes wird die Suche ausgeführt. Nach einer erfolgreichen Suche werden sämtliche Resultate ähnlich der Mapping-Liste auf der „Mappings View“-Seite aufgelistet. Wiederum kann der Benutzer mit den drei Symbolen auf der rechten Seite eines jeden Resultats eine bestimmte Aktion auslösen. Da die Symbole die gleiche Funktionalität aufweisen wie die der Mappings-Liste, seien sie hier nur kurz erwähnt. Das Lupen-Symbol zeigt ein neues Fenster mit zusätzlichen Informationen über das Mapping, das Zahnrad-Symbol öffnet das betreffende Mapping zur Bearbeitung auf der Editor-Seite, während das Papierkorb-Symbol nach einer Sicherheitsabfrage das gewünschte Mapping löscht.
Wie in diesem Kapitel etwas ausführlicher dargestellt wurde, besteht der Hauptzweck der Web-Applikation „RENE“ in der komfortablen und übersichtlichen Erstellung, Bearbeitung und Darstellung von Mappings zwischen Datenbanken. Leider konnte, nicht zuletzt aus Zeitgründen, nicht an allen Stellen reale Daten in die Web-Applikation eingefügt werden. So stellen beispielsweise die Informationsgruppen, die Informationsuntergruppen und die dazu gehörenden Entitäten reine Platzhalter dar. Auch real existierende Mappings können aktuell noch nicht in die Applikation importiert werden. An dieser Stelle sei jedoch erwähnt, dass unterschiedlichste Schnittstellen in der Web-Applikation bereits zur Verfügung stehen. Somit sollte es ein Leichtes sein, diese Schnittstellen dahingehend anzupassen beziehungsweise zu verbinden, damit Zugang zu realen Daten besteht. Des Weiteren darf die Tatsache nicht ausser Acht gelassen werden, dass zwar die Überprüfung von Mappings eine Schlüsselfunktion dieser Applikation darstellt, sie jedoch die Mappings ausschliesslich auf syntaktische Fehler überprüfen kann, nicht jedoch auf semantische. Aber auch hier ist eine passende Erweiterung aufgrund des gewählten Designs von „RENE“ in einer sinnvollen und vertretbaren Zeitdimension möglich.

4. Gesammelte Erfahrungen

Nachdem im vorangegangen Kapitel näher auf die technischen Details sowie auf deren Umsetzung in der Web-Applikation „RENE“ eingegangen wurde, soll dieses Kapitel dazu dienen, unsere Erfahrung, welche wir während der Durchführung des Software-Projektes sammeln durften, näher zu beschreiben. Dabei soll bewusst der Fokus auf drei wichtige Erfahrungsaspekte gelegt werden, welche, wie wir denken, uns in Zukunft am meisten von Nutzen sein werden.
Während wir in diesem durchaus anspruchsvollen Software-Praktikum unser Wissen und unsere Erfahrung betreffend des Software Engineerings wie gehabt einsetzen und auch erweitern konnten, mussten wir uns das erste Mal selbstständig mit dem Thema des Requirements Engineering auseinandersetzen, noch bevor wir eine einführende Vorlesung, welche von der Universität Zürich erst auf Master-Stufe angeboten wird, besuchen konnten. Dabei gingen wir von der Vorstellung aus, dass bereits erhobene und auch zukünftige Requirements eine Applikation betreffend, oder ganz allgemein ein Software-Projekt betreffend, stabil sind. Wir dachten also, dass sich Requirements in Zukunft nicht ändern und falls sie es doch tun, dann nur in Detailaspekten. Im Nachhinein erscheint uns dieser Gedanke etwas naiv. Trotzdem beschäftigte uns anfangs die Erhebung und die Dokumentation von (sich ändernden) Requirements der Web-Applikation „RENE“ ziemlich intensiv. Dabei fiel die Änderung von bereits erstellen Software-Artefakten nicht sonderlich ins Gewicht, da zu diesem Zeitpunkt kaum Code geschrieben worden war. Ab Mitte des Praktikums konnten wir uns jedoch zunehmend der Implementierung widmen. Dass sich also bereits erhobene Requirements ändern können und dies auch nachdem man mit der eigentlichen Implementierung begonnen hat, ist eine wichtige Erfahrung, welche wir machen durften.
Ein weiterer Erfahrungsaspekt, den wir an dieser Stelle erwähnen möchten, ist der Aspekt der zeitlichen Planung beziehungsweise des Abschätzens des zeitlichen Aufwands eines so genannten Meilensteines. Nicht allzu selten kam es in diesem Praktikum vor, dass wir Mühe hatten, den Zeitaufwand einer Anforderung oder eines fertigzustellenden Artefaktes präzise einzuschätzen. Oft genug trafen wir eine falsche Einschätzung der Situation. Uns ist durchaus bewusst, dass das Schätzen des zeitlichen (und somit auch des finanziellen) Aufwandes in einem IT-Projekt überaus schwierig ist, mit Hilfe dieses Praktikums konnten wir jedoch unsere Erfahrung dahingehend sinnvoll erweitern.
Die Übernahme eines kompletten Software-Projektes ist die dritte und letzte Erfahrung, auf welche wir hier eingehen möchten. Von der ersten zündenden Idee, über Requirements, Dokumentation, Implementation, Tests, bis hin zu der Abnahme der fertigen Artefakte durften wir die Entstehung einer Applikation nicht nur miterleben, sondern auch massgebend beeinflussen und prägen. Wir verstanden sehr schnell, dass an vielen Stellen im Entwicklungsprozess (teils schwierige) Entscheidungen gefällt werden müssen, welche selten eindeutig und offensichtlich sind, da viele Probleme auf die unterschiedlichsten Arten gelöst werden können. Neben technischen Aspekten spielt immer auch eine persönliche Note mit, die schliesslich das Aussehen und die Funktionsweise einer Software bis zu einem gewissen Grad prägt. Ein Stück Erfahrung konnten wir gewinnen, damit uns solche Entscheidungen in Zukunft etwas leichter fallen.
Abschliessend sei an dieser Stelle erwähnt, dass uns das Software-Projekt, welches wir in der Credit Suisse erfüllen durften, rückblickend auf unser Studium wohl am meisten zusätzliche, wertvolle Erfahrung gebracht hat.

5. Ausblick

In diesem letzten Kapitel soll nun ein Ausblick gewagt werden. Es soll aufgezeigt werden, welche Erweiterungen für die Web-Applikation „RENE“ einerseits möglich, andererseits sinnvoll sind.
Wie bereits in den vorangegangenen Kapiteln erwähnt, benutzt die Web-Applikation „RENE“ an vielen Stellen Platzhalter und keine realen Daten. Eine Ausnahme bilden hier die Domänen- und die Applikationsliste, welche über eine Schnittstelle bereits heute mit „richtigen“ Daten versorgt werden. Doch auch für das Einlesen von Informationsgruppen, Informationsuntergruppen, Entitäten und natürlich den eigentlichen Mappings existieren verschiedenste Schnittstellen in „RENE“, welche genutzt werden können, um Datenbanken mit realen Daten zu integrieren. Diese Erweiterung ist also ein wichtiger Schritt hin zu einer nutzbaren und hilfreichen Web-Applikation innerhalb der Credit Suisse.
Ein weiterer sinnvoller Ausbau der Applikation betrifft die (automatische) Überprüfung von Mappings. In der aktuellen Version können neue sowie bestehende Mappings nur aufgrund ihrer Syntax auf Korrektheit geprüft werden. Das bedeutet schliesslich, dass durchaus Mapping-Regeln erzeugt werden können, welche sich auf nicht existierende Entitäten beziehungsweise Attribute, oder auf sich selber beziehen. Dies ist natürlich nicht im Sinne des Erfinders. Daher wäre eine Erweiterung, welche es dem Benutzer erlaubt, Mappings zusätzlich auf ihre semantische Korrektheit hin zu überprüfen, sehr sinnvoll. Es ist offensichtlich, dass solch eine Funktionalität sehr wohl ein gewisser Entwicklungsaufwand bedeutet, die dabei gewonnene Effizienz bei der Arbeit mit Mappings darf jedoch nicht ausser Acht gelassen werden.
Eine dritte und letzte Erweiterung, welche wir hier vorschlagen möchten, bezieht sich auf die eigentliche Editor-Seite. Aktuell ist der Benutzer mit verschiedenen Gruppen von Schaltflächen, ähnlich eines Taschenrechners, konfrontiert. Diese Benutzeroberfläche zeigt zwar zu jedem Zeitpunkt alle möglichen Anweisungen, ist jedoch nicht kontext-sensitiv. In einer zukünftigen Version wäre also die Interpretation des aktuellen Kontextes ein Schritt in die richtige Richtung. Dabei fiele das Gitter an Anweisungs-Schaltflächen weg, damit mehr Platz für das Editor-Feld bliebe. In diesem könnte nun an der Position des Cursors ein kleines Hilfsfensters (beispielsweise implementiert als Pop-Up-Dialog) angezeigt werden, welches dem Benutzer zum aktuellen Kontext passend sinnvolle Vorschläge unterbreitet, zum Beispiel in Form von Erklärungen, der Anzeige des korrekten Syntax, oder mit ergänzenden Code-Beispielen. Ein einfacher Tastendruck würde genügen, um den Vorschlag einzufügen, so wie man es aus vielen modernen Programmierumgebungen kennt.
Alle diese Vorschläge mögliche Erweiterungen die Web-Applikation „RENE“ betreffend führen unserer Meinung nach weiter zu einem sinnvollen, effizienten, mächtigen, aber dennoch intuitiv zu bedienenden Werkzeug. Ein Werkzeug, welches der Credit Suisse in Zukunft hoffentlich wertvolle Dienste leisten wird.