HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

IFS INSTITUTE FOR SOFTWARE

Computer-Based Training mit Quiz und Reengineering eines Desktop Konverters OGR

Studienarbeit in Informatik
Bachelorstudium
an der Hochschule für Technik Rapperswil
der Fachhochschule Ostschweiz

Autor:
David Tran
13. Juni 2014

Betreuer:
Prof. Stefan Keller

Erstellung 17. Februar 2014

Letzte Aktualisierung 13. Juni 2014

Autor: David Tran

Betreuer: Prof. Stefan Keller (sfkeller@hsr.ch), IFS-HSR

Weitere Informationen im Web:

http://www.hsr.ch/

Computer-Based Training mit Quiz und Reengineering des Desktop-Konverters OGR

Studienarbeit von David Tran

Abteilung Informatik, Frühjahrssemester 2014

Ausgangslage

Die Qt-Software ist ein modernes Benutzerschnittstellen-Framework und eine mächtige Cross-Plattform-Entwicklungsumgebung für C/C++. Diese soll in dieser Arbeit auf zwei Arten genutzt werden.

Aufgabenstellung

Die Aufgabenstellung teilt sich in zwei unabhängige Teile, die beide mit Qt zu tun haben: Im ersten Teil wird Qt als Anwender genutzt, indem Quiz-Texte geschrieben werden, die in QGIS („Quiz Plugin“) und Moodle abgespielt werden. QGIS ist ein Desktop-Geoinformationssystem und Moodle ein webbasiertes Learning-Management-System, das an der HSR eingesetzt wird.

Im zweiten Teil geht es um die Aktualisierung einer Qt-basierten Software, einem Desktop-Konverter namens OGR2GUI. Dies ist ein GUI-Frontend zu einem bewährten Vektordaten-Konverter OGR als eigenständige Desktopapplikation. Nutzer von OGR2GUI können damit grosse Mengen von Geodaten in unterschiedliche Formate und Koordinatensystem konvertieren, bzw. transformieren. Dies würde die Arbeit vieler Nutzer vereinfachen, wird doch die veraltete Webseite von durchschnittlich 1500 Besucher pro Monat aufgerufen.

Aufgaben

- **Teil 1**. Selbstlern-Fragen zu DBs erstellen (u.a. Multiple Choice, Lückentext) mit Blick darauf, dass Dbs1 neu im ersten Semester Bachelor Informatik angeboten wird (u.a. auch für Maschinentechnik-Studenten).
  - Ziel: Quiz-Fragen zum Dbs1-Stoff im GIFT-Textformat erstellen (http://docs.moodle.org/26/en/GIFT_format) editiert mit Programmiereditor (z.B. Notepad++)
  - Abspielbar u.a. mit Moodle und QGIS Quiz.
  - Grundlage ist u.a. Datenbank-Buch
- **Teil 2**. „OGR2GUI“. Die existierende veraltete Version 0.6 (2009, www.ogr2gui.ca) dient als Ausgangslage, welche auf eine Version 0.7 aktualisiert und erweitert werden soll und zwar wie folgt:
  1. Auf aktuellste Qt-Version 5.2.1 migrieren.
  2. Auf aktuellste OGR-Version (GDAL/OGR/ 1.11) migrieren.
  3. Wenn möglich mit C++/11 realisieren (sonst C/C++ oder nur C).
  4. Unit-Tests ergänzen.
  5. Zusätzliche ausgewählte Formate (z.B. GeoPackage) und Webdiente (z.B. WFS) einbauen.
  6. Zusätzliche Parameter erlauben, namentlich „dataset creation option“, „layer creation option“ und „clipping“.
  7. Das GUI entsprechen ergänzen.

Vorgaben/Rahmenbedingungen

- Die Software läuft unter Windows (x64, wenn möglich auch x32).

- Der Student entscheidet sich nach Rücksprache mit dem Betreuer für eine SW-Entwicklungsmethodik. Die Meilensteine werden mit dem Betreuer und allfälligen Projektpartnern vereinbart.
- User Interface, Code, Kommentare und Versionsverwaltung, README (inkl. Installationsanleitung) sind in Englisch. Alles andere ist deutsch.
- Ansonsten gelten die Rahmenbedingungen, Vorgaben und Termine des Studiengangs bzw. der HSR.
- Ergänzung vom 17.3.2014: Der Abgabetermin wird neu auf 13.06.2014 festgelegt, d.h. um 2 Wochen verlängert.

Inhalt der Dokumentation

- Die fertige Arbeit muss folgende Inhalte haben:
  1. Abstract, Management Summary, Aufgabenstellung
  2. Technischer Bericht
  3. Projektdokumentation
  4. Anhänge (Literaturverzeichnis, CD-Inhalt)
- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind (einheitliche Nummerierung).
- Die Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen (nicht ausschliesslich Wikipedia-Links auflisten).
- Dokumentation des Projektverlaufes, Planung etc.
- Weitere Dokumente (z.B. Kurzbeschreibung, Poster als PDF) gemäss Vorgaben des Studiengangs und gemäss Absprache mit dem Betreuer.

Form der Dokumentation

- Bericht (Struktur gemäss Beschreibung) gebunden (2 Exemplare)
- Alle Dokumente und Quellen der erstellten Software auf CD; CD's sauber angeschrieben (2 Ex. Betreuer, 1 Ex. Abteilung).

Bewertungsschema

Es gelten die üblichen Regelungen zum Ablauf und zur Bewertung der Studienarbeit des Studiengangs Informatik mit besonderem Gewicht auf moderne Softwareentwicklung wie folgt:

- Projektorganisation (Gewichtung ca. 1/5)
- Bericht, Gliederung, Sprache (Gewichtung ca. 1/5)
- Inhalt inkl. Code (Gewichtung ca. 2/5)
- Gesamteindruck inkl. Kommunikation mit Industriepartner (Gewichtung ca. 1/5) (*)

(*) Es ist zwingender Bestandteil der Arbeit, dass eine lauffähige neue Version 0.7 von OGR2GUI abgeliefert wird.

Beteiligte

Student

David Tran

Projektpartner

(Mathieu Lahaye, Inventis, Québec, Kanada)

Betreuung HSR

Verantwortlicher Dozent: Prof. Stefan Keller, IFS-HSR (sfkeller@hsr.ch)

Erklärung

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.
- das ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum: 10. Juni 2014

David Tran: David Tran

Vereinbarung

Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit *Computer-based Training mit Quizzes und Reengineering eines Desktop Konverters* von *David Tran* unter der Betreuung von *Prof. Stefan Keller* geregelt.

Urheberrecht

Die Urheberrechte stehen dem Student zu.

Verwendung

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

Rapperswil, den 25.02.2014 David Tran

David Tran

Rapperswil, den 25.2.14 Stefan Keller

Prof. Stefan Keller

Abstract

Diese Arbeit befasst sich mit zwei Aufgaben, die unabhängig voneinander sind.

Im ersten Aufgabenteil geht es darum, ein Quiz für eine Einführung in Datenbanksysteme zu erstellen. Damit können Lernende im Selbststudium ihre Kenntnisse überprüfen. Es werden drei Fragetypen unterschieden: Multiple-Choice-, Zuordnungs- und Lückentext-Fragen.

Der zweite Aufgabenteil befasst sich der programmatischen Erweiterung eines Desktop Daten-Konverters namens OGR2GUI Version 0.6. Diese in C++/C geschriebene Software wurde 2009 das letzte Mal aktualisiert. OGR2GUI erweitert nun das Command-Line Werkzeug OGR (vermutlich Version 1.5), um ein grafisches User-Interface (GUI) auf der Basis von Qt. Trotz der veralteten Version von OGR erfreut sich laut den Autoren OGR2GUI bis heute grosser Beliebtheit. Das Ziel dieses Teils ist daher, einerseits die aktuellsten Versionen von OGR und Qt in OGR2GUI zu integrieren, sowie das Werkzeug um weitere OGR-Parameter zu erweitern.

Das Quiz wurde in thematische Kategorien eingeteilt und im textuellen GIFT-Format codiert. Das Ergebnis wurde mit Moodle und mit einem neuen QGIS Quiz-Plugin getestet, das auch offline verwendet werden kann. Moodle würde noch weitere Quiz-Fragetypen kennen, womit es noch interessanter gemacht werden könnte.

Als Resultat des zweiten Teils ist OGR2GUI Version 0.7 entstanden. Es wurde mit den Bibliotheken Qt 5.2.1 und GDAL 1.11 aktualisiert. Neu ist ein Webservice („Web Feature Service“) implementiert und es gibt ein freies Eingabefenster für bestimmte Optionen. Es ist geplant, dass die neue OGR2GUI Version wieder auf der Webseite www.ogr2gui.ca veröffentlicht wird.

Folgende Verbesserungen von OGR2GUI wären denkbar: Weitere Sprachen des GUI (Internationalisierung), „Drag and Drop“ von Dateien sowie zusätzliche OGR Formate, die bisher nicht vom GUI unterstützt werden.

Management Summary

- Teil 1: Computer-Based Training mit Quiz
- Teil 2: Reengineering eines Desktop Konverters OGR

Ausgangslage

Computer-Based Training mit Quiz
Moodle ist die E-Learning Plattform für das Quiz. Das Modul Datenbankmodul 1 ist die Grundlage dafür. Drei Fragetypen kommen in der Fragensammlung vor:

- Multiple-Choice-Fragen
- Zuordnungsfragen
- Lückentextfragen

Im Modul dient zur Unterstützung das Buch von Faeskorn-Woyke: Datenbanksysteme - Theorie und Praxis mit SQL2003, Oracle und MySQL. Mit Hilfe dieses Buch sind Fragen zu folgenden Kategorien erstellt:

- Datenmodellierung (Domain Modell)
- UML-Klassendiagramm, ER-Modell, Martin-Notation/Krähenfussdiagramm
- Relationales Modell (inkl. Abbildung Domain Modell in relationales Modell)
- Relationale Algebra und Normalisierung
- SQL Data Definition Language (DDL, PostgreSQL)
- SQL Data Manipulation Language (DML, PostgreSQL)
- SQL Security (DCL, PostgreSQL)
- Transaktionen
- Indexe und Optimierung
- JDBC
- OR Mapping
- PostgreSQL

Reengineering eines Desktop Konverters OGR
Die Firma Inventis hat OGR2GUI bis zum Release 0.6 entwickelt. Damit können Geodaten in andere Formate als die Eigene konvertiert werden. Die Geodaten müssen als Dateien, Dateien in einem Verzeichnis oder in einer GIS kompatible Datenbank vorliegen. OGR2GUI bietet in einem Dialogfenster an, eine Datenbankverbindung mit kompatiblen Datenbanksystemen herzustellen.

In der Abbildung 3 zu sehen, ist als Quelle eine geöffnete ESRI Shapefile, welche die Projektion 4269, d.h. NAD83 hat. Mit der Query „SELECT * FROM route“ sind alle Tabellen ausgewählt. Das Ziel ist die Datenbank *postgis* in PostgreSQL/PostGIS und die Projektion ändert mit 4714, d.h. Bellevue.

Vor der Konvertierung ist im Ausgabefenster, die gesamte Anweisung nochmals dargestellt. Mit „Execute“ startet die Konvertierung und der Statusbalken beginnt zu laufen.

Vorgehen, Technologien

Computer-Based Training mit Quiz

Moodle ist eine Lernplattform, die diese Quiz Fragensammlung importieren/exportieren kann. Das GIFT-Format ist für die Fragentypen am ehesten geeignet. Das Format ist eine eigenständige Sprache und die Syntax ist einfach erlernbar. Ein Texteditor genügt um die Textdateien zu bearbeiten. Ein Beispiel für eine Lückentextfrage:

```
Mein Geburtstag ist am {
        ~1.
        =29.
} Februar.
```

Die falsche Antwort ist 1. und 29. ist richtig. Das Symbol vor der Auswahl kennzeichnet die Aussage als =richtig/~falsch und die Klammern {} zeigen die Lösungsmenge an.

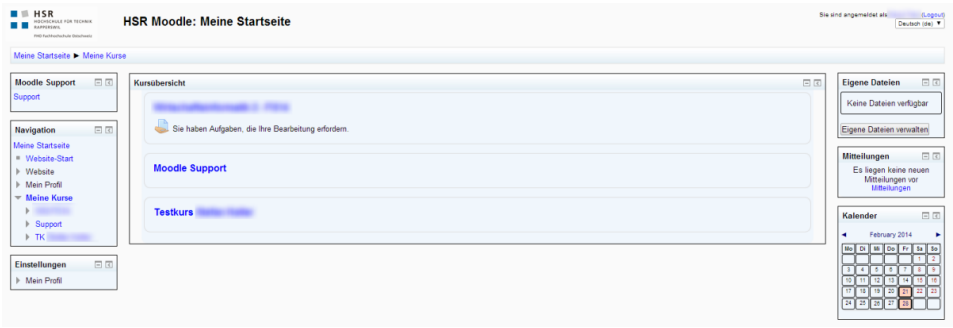


Abbildung 1 Moodle

In die Moodle Plattform muss die Fragensammlung als GIFT-Format. Die Kategorien der Fragensammlung müssen dann noch in einen Kurs eingefügt sein. Sobald der Kurs erstellt ist, kann das Quiz beginnen. Die Auswertung macht Moodle und zeigt beim Resultat die richtigen/falschen Auswahl mit der Musterlösung an.

Reengineering eines Desktop Konverters OGR

Der Rational Unified Process (RUP) ist das Modell für die Softwareaktualisierung. Der Plan zeigt Phasen/Iterationen und Meilensteine. Zuerst ist der alte Quelltext analysiert und der Aufwand geschätzt. Die Phasen/Iterationen geben eine Übersicht über den Ablauf über das restliche Semester und die Meilensteine beschreiben alles noch etwas genauer. Die Arbeitspakete sind in einer separaten Excel-Datei, in der auch die Zeiterfassung steht.

Der Übergang vom Qt 4.5 nach Qt 5.2 verläuft reibungslos. Der Webservice Web Feature ervice (WFS) ist implementiert. Ein Dialogfenster öffnet, wenn Webservice ausgewählt ist und eine WFS-Verbindung aufgebaut werden soll. Eine Erfolgs-/Fehlermeldung zeigt den Status der Verbindungsaufbau an. Ogr2ogr ist in OGR2GUI für zusätzliche Optionen integriert.

Ogr2ogr ist ein Kommandowerkzeug, das zur Konvertierung von Geodaten gemacht ist. Voraussetzung ist die GDAL/OGR Bibliothek und alle dazugehörigen Abhängigkeiten. GDAL ist nur für Rasterdaten geeignet und OGR nur für Vektordaten. OGR ist ein Teil von GDAL und bietet vereinfachte Funktionen an. Ein ogr2ogr Beispiel, welches das räumliche Bezugssysteme (SRS) eines ESRI Shapefile per URL ändert:

ogr2ogr out Bahnhoefe.shp -t_srs http://spatialreference.org/ref/epsg/4326/

Ergebnisse

Computer-Based Training mit Quiz

Die GIFT-Format Dokumentation auf Englisch ist die Aktuellste und hat vollständige Beschreibungen zu den verwendeten Fragetypen. Es sind über 100 Fragen in der Fragensammlung erstellt. Jede Kategorie hat mindestens eine Frage. Getestet ist das Quiz mit Moodle und QGis Quiz Plugin.

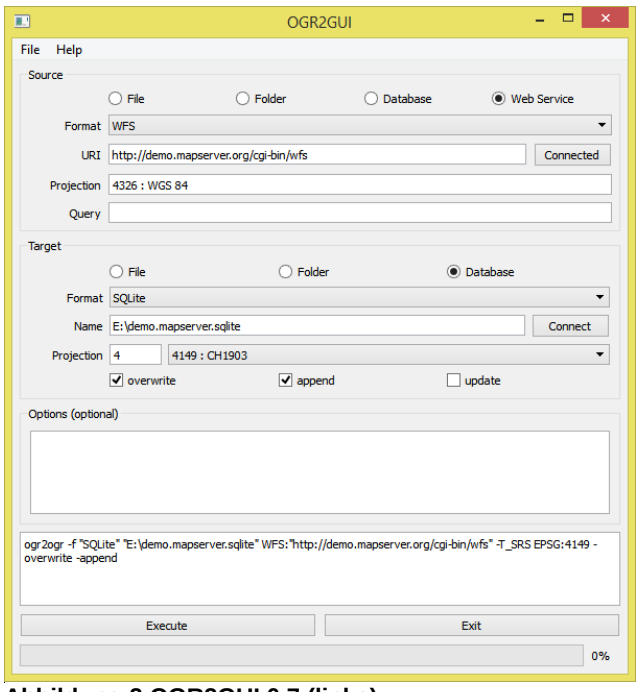
Es sind unschöne Fehler in Moodle, die zu finden sind:

- Maxmimal 255 Zeichen pro Zuordnung. Eine Fehlermeldung zeigt es ansonsten beim Import.
- Gleichlautende Zuordnungen kann Moodle nicht unterscheiden, so dass nach der Auswertung, das Ergebnis falsch-positiv sein kann.
- Moodle kürzt beim Import die Sätze beim zweiten Symbol „=“ ab, wenn das Symbol nochmals vorkommt.

Reengineering eines Desktop Konverters OGR

Die neue OGR2GUI Version ist implementiert. Diese OGR2GUI Version hat das Qt 5.2 Framework, die aktuellste GDAL 1.11 und VC10 als Compiler. Dieses Release kann man mit Microsoft Betriebssysteme ab Windows XP inklusive 64-bit als 32-bit Variante benutzen oder nur als 64-bit Variante. Zu empfehlen ist OGR2GUI x64 zu nehmen, weil während dem herunterladen von WFS Layern nicht abstürzt.

Eine neue GDAL muss man neu kompilieren. Zuerst ist der offizielle MinGW Toolchain getestet. Die Mingw-builds Distribution hat vorkompilierte Compiler für Intel/AMD Prozessoren. Die Datenbank MySQL kompiliert damit nicht, da es nur für Visual C/C++ 16.00 (VC10) kompatibel ist und ODBC kann nicht mit Mingw benutzt werden. Im Endeffekt blieb nur eine Visual C/C++ Version zu nehmen und viele Treiber kompilieren mit VC10.



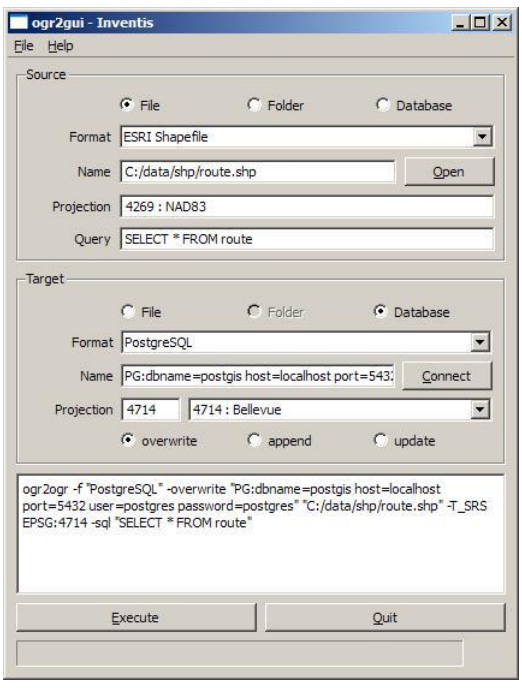


Abbildung 2 OGR2GUI 0.7 (links)

Abbildung 3 ogr2gui 0.6 (rechts)

Danksagung

Die Studienarbeit betreute Prof. Stefan Keller. Die Hochschule für Technik Rapperswil für die weiteren Praxis-Perspektiven.

Inhaltsverzeichnis

Kapitel I Technischer Bericht

1 Teil 1 Computer-Based Training mit Quiz

Der technische Bericht ist in zwei Teile aufgeteilt.

- Teil 1: Computer-Based Training mit Quiz
- Teil 2: Reengineering eines Desktop Konverters OGR

1.1 Einleitung

1.1.1 Problemstellung

Der erste Teil befasst sich mit einem Quiz zur Einführung in Datenbanksysteme, vor allem für das Modul Datenbanksysteme 1, und ähnliche Kurse. Das Quiz soll den Studenten und Kursteilnehmer helfen, ihre Datenbankkenntnisse im Selbststudium zu überprüfen und die Selbstkontrolle vereinfachen.

1.1.2 Aufgabenstellung

Es sind mindestens 100 Selbstlernfragen zu erstellen, wobei nur diese drei Fragetypen verwendet sind:

- Multiple-Choice-Fragen
- Zuordnungsfragen
- Lückentextfragen

Die Grundlage für die Fragensammlung ist das Buch von Faeskorn-Woyke[1]. Zu den folgenden Kategorien gehören die Selbstlernfragen:

- Datenmodellierung (Domain Modell)
- UML-Klassendiagramm, ER-Modell, Martin-Notation/Krähenfussdiagramm
- Relationales Modell (inkl. Abbildung Domain Modell in relationales Modell)
- Relationale Algebra und Normalisierung
- SQL Data Definition Language (DDL, PostgreSQL)
- SQL Data Manipulation Language (DML, PostgreSQL)
- SQL Security (DCL, PostgreSQL)
- Transaktionen
- Indexe und Optimierung
- JDBC
- OR Mapping
- PostgreSQL

1.1.2.1 Kann-Ziele

- Das Quiz kann das QGis Quiz Plugin vollständig verarbeiten.

1.1.2.2 Muss-Ziele

- Alle Selbstlernfragen im GIFT-Format sind mit Moodle kompatibel und es gibt keine Fehler, weder beim importieren noch exportieren.
- Die Selbstlernfragen und Lösungen sind auf Korrektheit und Rechtschreibung geprüft und sind richtig verarbeitet.
- Selbstkontrollen sind nur zu den vorgegebenen Kategorien und nur aus dem Buch.

1.1.3 Rahmenbedingungen

- ~~Abgabe der Kurzfassung an das Abteilungssekretariat am 30. Mai 2014.~~

- ~~Abgabe des Berichts ist spätestens am 30. Mai 2014 um 1700 Uhr. (*17.3.2014)*~~
- Abgabe der Kurzfassung/des Berichts um zwei Wochen verlängert bis am 13. Juni 2014 um 1700 Uhr.

1.1.3.1 Technologien

- Notepad++
- Moodle
- QGis Quiz Plugin

1.1.4 Vorgehen

1. Buch[1] lesen und Selbstlernlösungen in eine Textdatei schreiben.
2. GIFT-Format[2] lernen und die Selbstlernfragen zu den Fragetypen verteilen.
3. Das Quiz importieren/exportieren/testen mit der Moodle Plattform auf moodle.hsr.ch
4. Es sind über 100 Selbstlernfragen erstellt und in die Moodle Plattform importiert.

1.1.5 „Stand der Technik“

Moodle ist ein objektorientiertes Kursmanagementsystem, eine Lernplattform auf Open-Source-Basis. Die Software bietet die Möglichkeiten zur Unterstützung kooperativer Lehr- und Lernmethoden.[1]

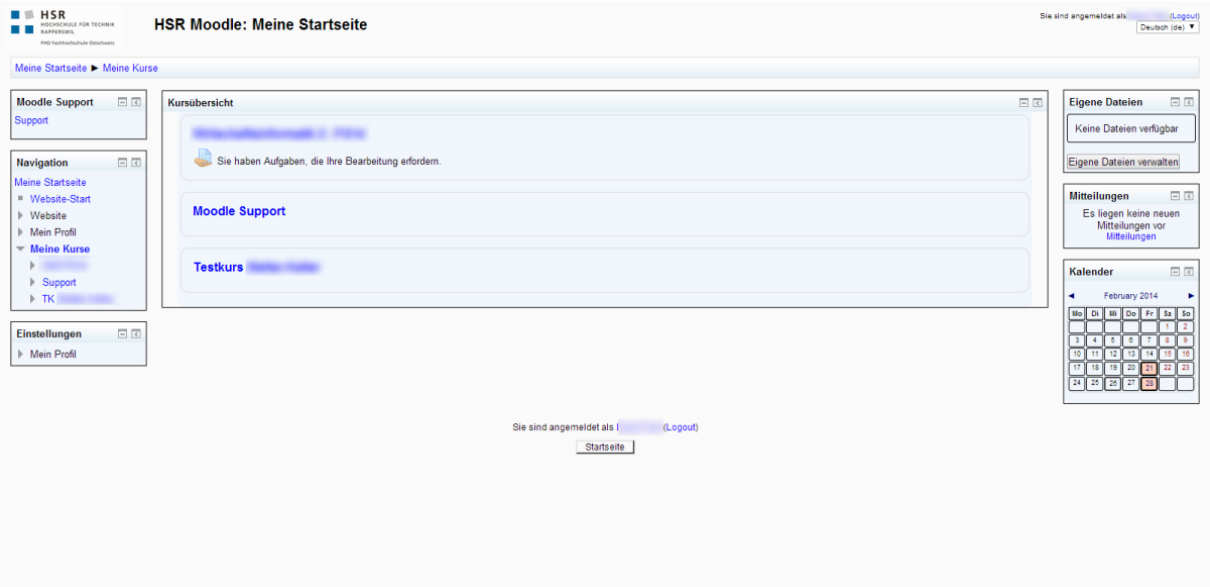


Abbildung 4 Moodle Startseite

1.1.5.1 GIFT-Format

Das GIFT-Format kann man in einer Textdatei (UTF-8) benutzen, die Moodle als Fragensammlung importiert. Das Format bietet insgesamt sechs Fragetypen an. In dieser Studienarbeit sind drei davon im Quiz verwendet und das wäre:

- Multiple-Choice-Fragen
- Lückentextfragen
- Zuordnungsfragen

Die Quiz kann in Moodle auch wieder als GIFT-Format exportiert werden. Damit ist es viel einfacher eine Fragesammlung zu erstellen, ändern und zu löschen.

[1] http://de.wikipedia.org/wiki/Moodle (25.4.14)

Formatierungssymbole:

Symbol	Verwendung
//	Kommentar
::Title::	Titel
=	Richtige Antwort
~	Falsche Antwort
#	Feedback
->	Zuordnung
{	Beginn der Antwortvorgaben
%50%	Gewicht von 50%
}	Ende der Antwortvorgaben

1.1.5.2 Beispiele

1.1.5.2.1 Multiple-Choice

```
Wer liegt in Grant's Grab in New York City? {
  =Grant
  ~Niemand
#Das stimmte 12 Jahre lang, aber Grant's sterblichen Überreste wurden dort 1897 bestattet.
  ~Napoleon
#Der wurde in Frankreich begraben.
  ~Churchill
#Der wurde in England begraben.
  ~Mutter Teresa
#Die wurde in Indien begraben.
}
```

1.1.5.2.2 Multiple-Choice mit mehreren Antworten

```
What two people are entombed in Grant's tomb? {
  ~%-100%No one
  ~%50%Grant
  ~%50%Grant's wife
  ~%-100%Grant's father
}
```

1.1.5.2.3 Lückentextfrage

```
Mahatma Gandhi's birthday is an Indian holiday on  {
   ~15th
   ~3rd
   =2nd
} of October.
```

1.1.5.2.4 Zuordnungsfragen

```
Ordnen Sie den Ländern die richtigen Hauptstädte zu. {
   =Kanada -> Ottawa
   =Italien  -> Rom
   =Japan  -> Tokio
   =Indien  -> Neu Delhi
}
```

1.2 Ergebnisse

1.2.1 Bewertung (Evaluation)

Die englische GIFT-Format Dokumentation (Version 2.6) ist aktuell und hat Beschreibungen und Beispiele zu Multiple Choice, Zuordnungsfragen, Lückentextfragen. Die auf Deutsch verfasste Dokumentation (Version 2.6) ist unvollständig und teilweise falsch, zum Beispiel ohne Lückentextfragenbeispiele und die dazugehörige Beschreibung, dass Moodle keine Lückentextfragen importieren kann, ist falsch. Zusammengezählt sind es über 100 Fragen. Zu jeder Kategorie sind Dateien mit Selbstlernfragen erstellt, die alle drei Fragetypen enthalten. Es sind Total mindestens 18 Zuordnungsfragen, 26 Lückentextfragen und 44 Multiple Choice Selbstlernfragen gestellt und es hat mindestens eine Frage zu jeder Kategorie. Die GIFT-Format Dateien sind fehlerfrei und ausschliesslich für Moodle und für das QGis Quiz Plugin.

1.2.2 Umsetzungskonzept

- Datenbanksysteme 1 ist als Grundlage für das Quiz gegeben.
- Kategorien sind vorgegeben und mit PostgreSQL-Befehle ergänzt.
- Moodle und GIFT-Format kennenlernen.
- Fragensammlung mit drei verschiedenen Fragetypen erstellen.
- PostgreSQL Dokumentation ist für zusätzliche Selbstlernkontrollen.
- Moodle und QGis Quiz Plugin zum testen verwenden.

1.2.3 Resultate: Bewertung und Zielerreichung

In Moodle hat die Zuordnungsfrage eine Zeichenbeschränkung: Beim Test mit dem Zeichen ‚M' sind maximal 255 Zeichen pro Zuordnung erlaubt. Bei längeren Zuordnungstexten zeigt Moodle beim Importversuch die Fehlermeldung „Fehler beim Schreiben der Datenbank" an. Die Zuordnungen können gleichlauten (zum Beispiel via kopieren und einfügen), aber

Moodle kann diese nicht auseinanderhalten, noch kommt eine Fehlermeldung, dass eine oder mehrere Zuordnungen gleich sind. Das Ergebnis zeigt dann, dass das Zuordnungspaar falsch ist. Mit dem Zeichen ‚=' gebe ich die richtige Lösung an, wenn aber noch ein ‚='-Zeichen in der richtigen Antwort steht, dann kürzt Moodle den Satz bis vor dem ‚=' ab. Das Ergebnis ist wiederum falsch, weil meine Antwort nicht mit der richtigen Antwort übereinstimmt.

Es braucht über 69 Stunden für Teil 1 Computer-Based Training mit Quiz um über 100 Selbstlernfragen zu erfinden und ins GIFT-Format umzuschreiben. Dazu kommen noch über 8 Stunden Tests mit Moodle und dem QGis Quiz Plugin dazu. Moodle importiert und exportiert alle Selbstlernkontrollen als Fragensammlung fehlerfrei. Alle Selbstlernfragen sind auf Rechtschreibung geprüft. Die Lösungen sind in Moodle vollständig dargestellt. Neu ist das QGis Quiz Plugin, das auch mit den GIFT Dateien getestet ist und mit allen drei Fragetypen klar kommt.

1.3 Schlussfolgerungen

1.3.1 Schlussfolgerungen und Ausblick

Das Modul Datenbanksysteme 1 und die darin festgelegten Kategorien dienen als Grundlage für die Selbstkontrolle und mit dem Buch von Faeskorn-Woyke[1] sind die Selbstlernfragen gestellt. Die Studenten vom Modul Datenbanksysteme 1 oder die Teilnehmer eines Kurses können ihr Selbststudium damit überprüfen. Mit Moodle oder QGis Quiz Plugin können die Dateien (GIFT-Format) importiert bzw. geöffnet werden. Moodle stellt die Dateien einer Kategorie in eine Fragensammlung, die dann in die Aktivität „Test" hinzugefügt werden. Es sind insgesamt 102 Selbstkontrollfragen und -lösungen zum Datenbanksysteme 1 Stoffumfang und als Fragetypen Multiple Choice, Zuordnungsfragen und Lückentextfragen abgefragt.

Das Thema Datenbanksysteme führt das Quiz ein. Es kann mit dem Modul Datenbanksysteme 2 und Oracle Datenbanken oder andere Datenbanken erweitert werden. Ebenso sind nur drei Fragentypen in diesem Quiz verwendet worden, aber Moodle unterstützt noch weitere Fragentypen und Formate, darum ist es auch hier ausbaufähig. Eine grössere Testgruppe kann das Quiz genauer testen und fehlerfrei machen.

1.3.2 Persönlicher Bericht

Ich habe vor dieser Studienarbeit noch nie von Moodle gehört, noch etwas darüber gelesen. Als Übungsplattform ist es sehr interessant. Die Fragensammlung kann ich offline mit Notepad++ erarbeiten und beim nächsten Internetzugang mit der Importfunktion ins Moodle laden. Das GIFT-Format kannte ich bisher nicht, aber die Syntax ist einfach zu lernen. Verschiedene Fragetypen machen es einfach, ein spannendes Quiz zu erstellen.

2 Teil 2 Reengineering eines Desktop Konverters OGR

Der technische Bericht ist in zwei Teile aufgeteilt.

- Teil 1: Computer-Based Training mit Quiz
- Teil 2: Reengineering eines Desktop Konverters OGR

2.1 Einleitung

2.1.1 Problemstellung

Im Jahr 2009 gab es die letzte Veröffentlichung von OGR2GUI 0.6, aber nach der Downloadstatistik von Sourceforge.net ist es immer noch eine sehr beliebte Anwendung, obwohl OGR2GUI technisch schon sehr veraltet ist.

2.1.2 Aufgabenstellung

Im zweiten Teil geht es darum das Tool OGR2GUI von Inventis mit dem aktuellsten Qt Framework 5.2.1 zu aktualisieren und um die neue Quelle „Webservice“ zu erweitern. Die Quelltexte sind mit dem aktuellsten C-Compiler kompiliert und sind im Git Repository abgelegt.

2.1.2.1 Kann-Ziele

- Mit WINE testen (Ubuntu 14.04).

2.1.2.2 Muss-Ziele

- GUI Komponente bleiben wie sie waren.
- Mit Qt 5.2.1 aktualisieren.
- Aktuellste GDAL/OGR Bibliothek hinzufügen.
- Eine neue Quelle „Web Service“ implementieren.
- Optioneneingabe implementieren.
- Mit Visual C++ 16.00 Toolchain kompilieren.
- Unit Tests.
- System testen mit Windows 8.1.

2.1.3 Rahmenbedingungen

- ~~Abgabe der Kurzfassung an das Abteilungssekretariat am 30. Mai 2014.~~
- ~~Abgabe ist spätestens am 30. Mai 2014 um 1700 Uhr. (*17.3.2014)*~~
- Abgabe des Berichts um zwei Wochen verlängert bis am 13. Juni 2014 um 1700 Uhr.

2.1.3.1 Software Engineering

OGR2GUI wird nach dem Rational Unified Process (RUP) aktualisiert. Die Iterationen:

- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment

Die Phasen:

- Inception
- Elaboration

- Construction
- Transition

2.1.3.2 Technologien

- Qt 5.2.1
- Qt Creator 3.1
- Visual Studio 2010

2.1.4 Vorgehen

1. Qt einarbeiten und die Entwicklungsumgebung kennenlernen.
2. OGR2GUI Quelltexte in Qt Creator importieren.
3. Fehlende Bibliotheken kompilieren.
4. Webservice als neue Quelle implementieren und testen.
5. Die Quelltexte mit MSVC (Microsoft Visual C++) Compiler kompilieren.

2.1.5 „Stand der Technik“

Das Desktoptool OGR2GUI 0.6 konvertiert und ändert Geodaten. Es ist eine grafische Benutzeroberfläche zur ogr2ogr Kommandozeile. Die Geodaten können per Datei/Ordner oder Datenbank eingelesen werden und werden wieder in eine Datei/Ordner oder Datenbank geschrieben, dabei kann man die Daten überschreiben (overwrite), anhängen (append) oder aktualisieren (update).

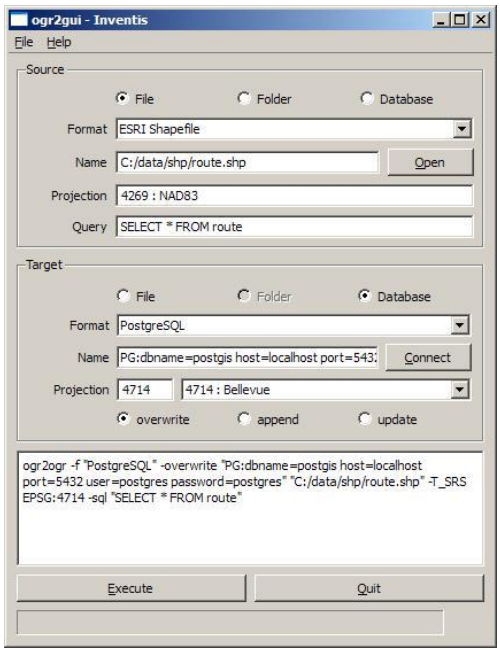


Abbildung 5 OGR2GUI v0.6

2.2 Ergebnisse

2.2.1 Bewertung (Evaluation)

Der offizielle MinGW Toolchain gibt es nur in 32-bit. Die alternative 64-bit Toolchain ist Mingw-w64. Mingw-w64 sind nur Quelltexte, darum gibt es verschiedene Distributionen. Es ist Mingw-builds, welche viele Bibliotheken/Treiber für GDAL (Geospatial Data Abstraction Library) fehlerfrei kompilieren kann. Getestet mit den Mingw Toolchains sind die GDAL

Versionen 1.9.2, 1.10.1. MinGW 32-bit kompiliert nur GDAL 1.9.2, ab 1.10.x geht nur noch Mingw-builds. Probleme entstehen mit den Datenbanken PostgreSQL 9.3.4 (kompiliert nur mit Mingw-builds), Oracle Instant Client 11/12 (GDAL erkennt nur die Headers), MySQL (nur mit VC10 kompatibel) und ODBC für die gibt es keine Treiber für Mingw Toolchains.

Die Auswertung der Toolchains mit GDAL und OGR2GUI:

OGR2GUI (OGR)	**X2GUI (ogr2ogr)**	**MSVC 10 (GDAL)**
OGR Programmiersprache: C	Binäres Gdal	OGR2GUI Quelltexte müssen portiert werden, weil MinGW *nix als Grundlage hat
Kein OCI (Instant Client Bibliothek wird von Gdal nicht erkannt)	C ist für mich eine unbekannte Programmiersprache	Keine Erfahrung mit MSVC
Kein ODBC (Treiber?)	Umgebungsvariablen müssen gesetzt werden	Gdal Headers und Libs für MSVC 2005/2008/2010 vorkompiliert.
Kein MySQL (kompiliert nicht mit Mingw-builds)	Individuelle Gdal-Versionen können zu es unbekannten Konflikte kommen	Ist OGR MSVC kompatibel? API Tutorial hat nur C und C++ Beispiele
Nur Gdal 1.9.2 kompiliert mit mingw32	Unit Tests?	
Gdal 1.10.1 braucht Mingw-builds	Einfachere Flag-Übergabe an ogr2ogr	
Libspatialite geht nur mit mingw32 und gdal 1.9.2	Formatevorgabe ist bereits festgelegt	
Gdal 1.11 wird bald Veröffentlicht	Neues Dialogfenster für ogr2ogr Pfad	
	Ogrinfo --formats Informationen müssen geparst und sortiert werden.	

OGR2GUI Version 0.6 ist mit Qt 4.5.2 erstellt. Es sind unvollständige Implementationen, zum Beispiel „overwrite, append, update“ sind nur in der View. Die Formateliste ist unvollständig, weil per Standard in der GDAL Bibliothek mehr Formate mitkompiliert sind. Eine unbekannte GDAL ist für OGR2GUI benutzt und ohne Header Quelltexte ausgeliefert.

Ausgebaut soll OGR2GUI mit optionalen Optionen. Dazu sind verschiedene Ansätze Untersucht. OGR2GUI mit OGR API Funktionen erweitern scheint auf den ersten Blick, die plausibelste Lösung zu sein. Leider fehlen in der OGR API Dokumentation einige Beispiele für die Optionen DSCO und LCO. GDAL API bietet dafür eine vollständige Dokumentation. Einen Wrapper zu schreiben, ist sehr aufwändig und kaum von Nutzen, wenn später Quelltexte dazukommen oder geändert sind und darum ist beschlossen worden, das nicht weiterzuverfolgen. Ein Qt-GUI mit ogr2ogr ist mit dem Quelltext umsetzbar und in OGR2GUI als Erweiterung integrierbar.

Um OGR2GUI mit Optionen zu programmieren, sind folgende Kriterien bewertet:

OGR2GUI Extended	**Ogr2ogrWrapper**	**Qt-GUI with ogr2ogr**
OGR_Dr_CreateDataSource http://www.gdal.org/ogr/ogr__api_8h.html#ac2b628f8ddc	ogr2ogr.cpp kompiliert nicht „einfach so“ und braucht darum einen Layer zwischen ogr2ogr.cpp und	Warum ein neues Qt-GUI? Ein neues Dialogfenster genügt für die Konfiguration.

674f72c798829c738bbdd	GDAL.	
OGR_DS_CreateLayer http://www.gdal.org/ogr/ogr__api_8h.html#a424d383a37fbeaea58acaea11717f320	Gdal-dev scheint keiner einen Ansatz für eine Implementation zu haben.	Logfenster kann für Optionen editierbar gemacht werden.
papszOptions: a StringList of name=value options. Options are driver specific, and driver information can be found	Der Gui-Quelltext bleibt gleich, nur die „Ogr“-Funktionen sind umgeleitet.	Der Gui-Quelltext bleibt gleich, nur die „Ogr“-Funktionen sind umgeleitet.
Mit Hilfe von ogr2ogr.cpp könnte es die Lösung für Optionen sein.	Logfenster als Ausgabefenster erweitern.	Logfenster als Ausgabefenster erweitern.
umsetzbar	Keinen brauchbaren Ansatz	umsetzbar
ungetestet	sehr schwierig	Neue Qt Quelltexte

2.2.2 Umsetzungskonzept

1. Die Quelltexte sind mit Qt 5.2.1 aktualisiert.
2. OGR2GUI Quelltexte sind mit den MSVC Toolchain kompiliert.
3. Die neue Quelle Webservice ist implementiert.
4. Optionen sind verarbeitet.
5. Binäre Dateien für die Softwareverteilung.

2.2.3 Resultate: Bewertung und Zielerreichung

OGR2GUI ist mit dem Qt 5.2.1 Framework aktualisiert. Qt 5 kann mit dem MSVC 2010 Toolchain umgehen. Damit sind die Quelltexte für die x86 und x64 Architekturen kompiliert. Der Webservice ist anhand eines Beispiels mit Web Feature Service (WFS) implementiert. Die Kennung ist ein Uniform Ressource Identifier (URI), damit verbindet WFS mit einem Geoinformationssystem (GIS) und das GIS sendet Geoinformationen als antwortet. MSVC ist schlussendlich das beste Toolchain um GDAL 1.11 zu kompilieren. Ausprobiert sind VC10 und 11 auch als MSVC 2010 und 2012 bekannt. Dabei kompiliert nur MSVC 2010 (x86 und x64) die meisten Bibliotheken als Treiber für GDAL. Probleme machen die Curl (libcurl.dll) und SSL (libeay32.dll, ssleay32.dll) DLLs, so dass WFS nicht mehr mit einem GIS verbindet. Lösen lässt sich das, in dem man diese DLLs mit dem, der vorherigen Versionen austauscht. Ein ogr2ogr-Wrapper ist sehr Zeit aufwändig und schwierig zu implementieren und nach jeder Aktualisierung von der ogr2ogr.cpp muss man es korrigieren, darum ist es nicht umgesetzt. Die OGR API mit C++ hat Warnungen, die aber die Kompilierung nicht behindern. Die Datenbankverbindung zu Oracle Datenbanken ist entfernt, weil nur ein älteres Oracle Datenbanksystem gratis herunterladbar ist.

2.3 Schlussfolgerungen

2.3.1 Schlussfolgerungen und Ausblick

Der Quelltext als Grundlage ist Ogr2gui Version 0.6 von Inventis. Es ist mit Qt 4 programmiert und benutzt eine GDAL Version 1.5 oder 1.6. Die neue OGR2GUI Version 0.7 kann Geodaten als Dateien oder Datenbank konvertieren. Nur beim Webservice WFS hat es Schwierigkeiten in verschiedene Formate zu schreiben. Ogr2ogr bietet an, Optionen als Parameter mitzugeben, darum ist dieses Feature auch in OGR2GUI mit mindestens einer Option (DSCO) implementiert.

Die neue Qt 5.3 wurde am 20. Mai veröffentlich. Im Vergleich mit Qt 5.2.1 gibt es keine grösseren Änderungen, die OGR2GUI betreffen, so dass OGR2GUI erst beim nächsten Meilenstein aktualisiert werden würde. OGR2GUI 0.7 ist darum nicht mehr damit aktualisiert. Die Bibliothek GDAL 1.11 ist mit Visual C++ 16.00 kompiliert. Der aktuellste Visual C++ Compiler hat die Version 18.00 (VC12) und ist mit Microsoft Visual Studio 2013 veröffentlicht. GDAL 1.11 ist um VC12 ergänzt worden. Alle Treiber müssen neu kompiliert und mit GDAL 1.11 getestet werden.

2.3.2 Persönlicher Bericht

OGR ist nicht vollkommen neu für mich, weil ich früher QGis gebraucht habe. Hingegen ist OGR2GUI mit Qt programmiert und GDAL benutzt eine API, die man mit C/C++ ansprechen soll. Bis dahin brauchte ich weder Qt noch C. Die Herausforderung bin ich aber gewachsen, darum fing ich als erstes an, mich in Qt und C einzuarbeiten. Qt Creator ist die Entwicklungsumgebung für Qt und dazu braucht es einen Compiler wie Mingw oder MSVC. Schlussendlich habe ich mich für VC10 entschieden, weil GDAL 1.11 damit kompiliert und auch die meisten Treiber. Nach und nach sind neue Funktionen ins OGR2GUI eingebaut, z.B. ein Webservice WFS, Eingabe für zusätzliche Optionen.

Kapitel II SW-Projektdokumentation

3 Teil 1: Computer-Based Training mit Quiz

3.1 Überblick

Hier folgen die Kapitel gemäss Software-Engineering-Vorgehen.

3.2 Vision

Im technischen Bericht (Teil 1).

3.3 Anforderungsspezifikation

3.3.1 Produkt Perspektive

Das Produkt ist ein Quiz mit 102 Fragen aus den Kategorien:

- Datenmodellierung (Domain Modell)
- UML-Klassendiagramm, ER-Modell, Martin-Notation/Krähenfussdiagramm
- Relationales Modell (inkl. Abbildung Domain Modell in relationales Modell)
- Relationale Algebra und Normalisierung
- SQL Data Definition Language (DDL, PostgreSQL)
- SQL Data Manipulation Language (DML, PostgreSQL)
- SQL Security (DCL, PostgreSQL)
- Transaktionen
- Indexe und Optimierung
- JDBC
- OR Mapping
- PostgreSQL

Mit Moodle oder dem QGis Quiz Plugin kann die Selbstkontrolle oder Selbststudium beginnen. Es werden Multiple-Choice-, Zuordnungs- und Lückentextfragen gestellt.

3.3.2 Produkt Funktion

Die Dateien sind im GIFT-Format und sind mit Moodle kompatibel und sind als Fragensammlung in Moodle importiert oder von Moodle aus exportiert. QGis hat ein Quiz Plugin, das ebenfalls mit den GIFT-Format Dateien umgehen kann. Das Quiz hat Multiple-Choice-Fragen, die eine oder mehrere richtige Antworten haben können, Lückentextfragen sind in kurzen Texten formuliert und hat immer nur eine Lücke, die mit mindestens einem Zeichen oder mehrere Wörter gelöst sind. Bei den Zuordnungsfragen sind jeweils Paare zu finden und zu jeder Zuordnung gibt es genau eine dazugehörige andere Zuordnung.

3.3.3 Benutzer Charakteristik

Die Studenten und Hörer an der Hochschule für Technik Rapperswil (HSR) können sich ab nächsten Herbst für das Modul Datenbanksysteme 1 (Dbs1) einschreiben. Das Modul wird ab nächsten Herbst ins erste Semester verschoben. Moodle ist darum ab nächsten Herbstsemester die Lernplattform für alle eingeschriebenen Studenten und Hörer. Für ähnliche Datenbanksysteme Kurse kann Moodle oder das QGis Quiz Plugin sinnvoll sein.

3.3.4 Einschränkungen

Die Fragensammlung ist mit dem Buch von Faeskorn-Woyke[1] erfunden. Die Dateien sind im GIFT-Format, welches von Moodle gewartet wird. Es sind mindestens 100 Quizfragen und nur die Fragetypen Multiple-Choice, Zuordnung und Lückentext sind benutzt.

3.3.5 Annahmen
Die GIFT-Format Dateien sind als Textdateien (UTF-8) gespeichert. Alle Quiz Fragen sind aus dem Buch Faeskorn-Woyke[1] und PostgreSQL-Befehle aus der Vorlesung und Übungen vom Modul Datenbanksysteme 1.

3.3.6 Abhängigkeiten
Das Quiz ist für die Moodle Lernplattform der Hochschule für Technik Rapperswil angepasst und auch mit dem QGis Quiz Plugin ist es fehlerfrei.

3.3.7 Use Cases
Detailbeschreibung sämtlicher Use Cases.

3.3.7.1 Use Case Diagramm

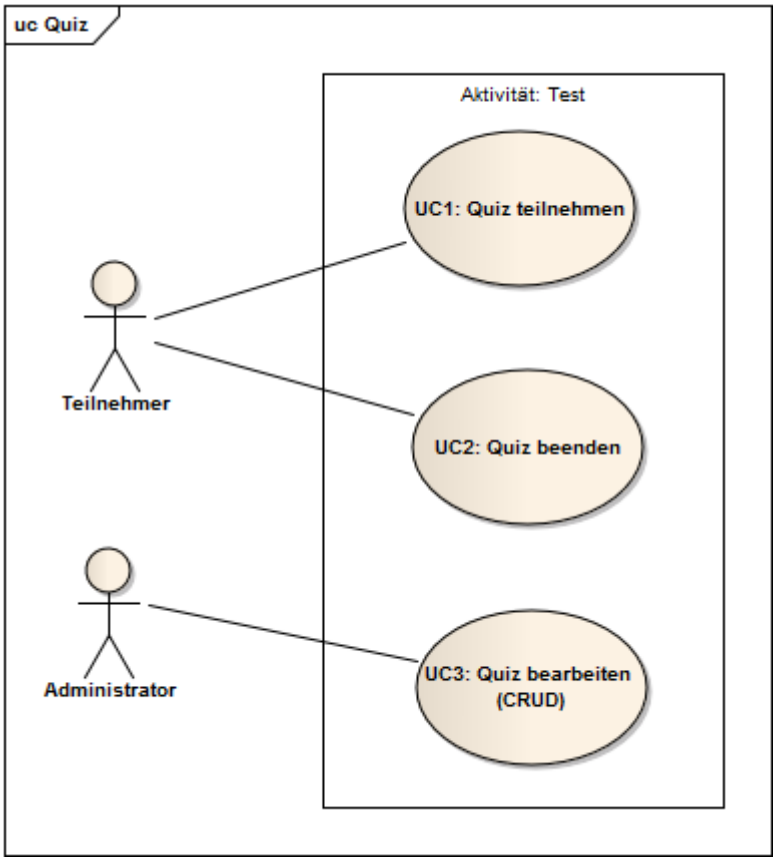


Abbildung 6 Use Case

3.3.7.2 Aktoren & Stakeholder
- Moodle als System:
  - Teilnehmer:
    - Kein direkter Actor für das Quiz.
    - Ziel: Kann Tests (Quiz) teilnehmen und beenden lassen.
  - Administrator:
    - Primary Actor
    - Ziel: Kann Quiz bearbeiten (CRUD) und importieren/exportieren.
- Aktivität „Test“ als System:
  - Teilnehmer:
    - Primary Actor

- Ziel: Kann Tests (Quiz) teilnehmen und beenden lassen.

3.3.7.3 Beschreibungen (Brief)

Alle Use Cases in einzelnen Kapiteln im Brief Format beschrieben.

3.3.7.3.1 Quiz teilnehmen

Der Teilnehmer kann eine Vorschau starten. Bei jeder Frage, kann der Teilnehmer eine Lösung eingeben. Jede vollständig gelöste Frage ist in der Test-Navigation farblich markiert. Ist die Frage ungelöst, bleibt das Feld in der Test-Navigation gleich wie vorher. Korrekturen sind gestattet.

3.3.7.3.2 Quiz beenden

Das Quiz kann der Teilnehmer nach der letzten Frage mit einer Abgabe beenden oder wenn es vom Teilnehmer gewünscht ist, kann es auch während dem Versuch beendet werden. Das Quiz kann dann nicht mehr bearbeitet werden. Das Resultat zeigt die richtig gelösten Fragen mit der Farbe grün an, bei nur teilweise richtigen Antworten, ist es gelb und die Farbe rot ist für die falsche Antwort.

3.3.7.3.3 Quiz bearbeiten

Falls neue Fragen importiert oder exportiert werden möchte, muss es von einem Administrator ausgeführt sein. Die Fragensammlung kann er aktualisieren und wenn die Fragen nicht mehr gebraucht werden, kann er diese löschen.

3.3.7.4 Beschreibungen (Fully Dressed)

Spezielle und wichtige Use Cases in einzelnen Kapiteln im Fully Dressed Format beschrieben.

3.3.7.4.1 Quiz teilnehmen

Scope	Quiz
Level	user-goal
Primary Actor	Teilnehmer
Stakeholder and Interests	Möchte ein Selbststudium/eine Selbstkontrolle beginnen.
Preconditions	Fragensammlung ist importiert.
Success Guarantee	Lösungen sind vollständig.
Main Success Scenario	1. Der Teilnehmer startet eine Vorschau. 2. Der Teilnehmer kann alle Fragen versuchen.
Extensions	Es sind keine Fragen im System.
Special Requirements	Kenntnisse in Datenbanksysteme.
Technology and Data Variations List	Keine.
Frequency of Occurrence	Jedes Mal wenn ein Quiz gestartet wird.
Miscellaneous	Keine.

3.3.7.4.2 Quiz beenden

Scope	Quiz
Level	user-goal
Primary Actor	Teilnehmer
Stakeholder and Interests	Möchte den Versuch beenden.
Preconditions	Der Versuch ist bereits gestartet.
Success Guarantee	Der Versuch muss nach der letzten Frage abgegeben oder während dem Versuch beendet

	werden.
Main Success Scenario	Der Teilnehmer beendet den Versuch.
Extensions	1. Die Lösung ist nicht im System. a. Ein Administrator muss die Frage bearbeiten. 2. Die Lösung ist falsch. a. Ein Administrator muss die Frage bearbeiten.
Special Requirements	Kenntnisse in Datenbanksysteme.
Technology and Data Variations List	Keine.
Frequency of Occurrence	Jedes Mal wen nein Quiz beendet wird.
Miscellaneous	Keine.

3.4 Design (Entwurf)

3.4.1 Systemübersicht

Folgende Auflistung zeigt die Formatierungssymbole:

Symbol	**Verwendung**
//	Kommentar
::Title::	Titel
=	Richtige Antwort
~	Falsche Antwort
#	Feedback
->	Zuordnung
{	Beginn der Antwortvorgaben
%50%	Gewicht von 50%
}	Ende der Antwortvorgaben

3.4.2 Deployment

Alle 12 Textdateien sind in GIFT-Format geschrieben und sind kompatibel mit Moodle von der HSR und mit dem QGis Quiz Plugin v0.1.

3.5 Implementation (Entwicklung)

Bei allen drei Fragetypen (Multiple-Choice, Zuordnung, Lückentext) braucht es etwas Text für die Quizfrage. Ein Beispiel:

```
Wer liegt in Grant's Grab in New York City? Grant
```

Um eine Multiple-Choice-Frage zu formulieren, werden als erstes die Frage und die Antwort separiert. Mit dem Symbol „{“ nach dem Fragezeichen sind alle möglichen richtigen „=“ oder falschen „~“ Antworten aufzulisten und mit dem Symbol „}“ wird die Auswahl abgeschlossen.

```
Wer liegt in Grant's Grab in New York City? {
        =Grant
        ~Niemand
        ~Napoleon
        ~Mutter Teresa
}
```

Wenn noch Feedback angezeigt werden soll, ist es nach der Antwort mit dem Symbol „#“ zu beginnen. Bei mehreren richtigen Antworten gibt man den Antworten ein Gewicht. Lückentextfragen sind den Multiple-Choice-Fragen mit einer Antwort sehr ähnlich, aber haben nach dem Symbol „}“ immer den restlichen Teil der Antwort. Zuordnungsfragen beginnen nach Beginn der Antwortvorgaben mit einem „=“ und das Paar wird mit der Zuordnung „->“ zugewiesen.

```
xyz. {
        =a -> b
        =c -> d
}
```

3.6 Test

3.6.1 Voraussetzungen

- Moodle (mit Administratorrechte)
- QGis Quiz Plugin
- Quiz

3.6.2 Vorbereitungen

Zu jede Datenbanksysteme 1 Kategorie ist eine Moodle Kategorie zu erstellen. Die Quiz Dateien müssen als Fragensammlung in Moodle importiert sein. Die Fragen in der Fragensammlung können mit der Vorschau angeschaut werden. Ebenso kann es in ein neues Thema als Aktivität „Test“ hinzugefügt werden und dann das Quiz als Versuch starten.

3.6.3 Systemtest

Beschreibung der einzelnen Tests der Use Cases.

Use Case	Beschreibung
UC1	1. Vorschau starten. 2. Alle Fragen mit den Lösungen lösen. 3. Überprüfen, ob die Fragen nach den Vorgaben gelöst werden kann.
UC2	1. Vorschau starten. 2. Alle Fragen mit den Lösungen lösen. 3. Überprüfen, ob die Lösungen vom System richtig verglichen worden sind. 4. Überprüfen, ob die Lösungen vom System richtig gespeichert wurden.
UC3	1. Dateien importieren und exportieren.

3.6.4 Angaben zur Durchführung

Es hat zwei vollständige Durchführungen gebraucht um das Quiz fehlerfrei zu machen. Bei der ersten Durchführung ist Moodle als Testumgebung und ist als Versuch zu starten und alle Fragen richtig zu lösen. Das QGis Quiz Plugin sind alle Fehler direkt dem Betreuer und Entwickler als Nachricht mit einer Problembeschreibung gemeldet. Bei der zweiten Durchführung ist Moodle und QGis Quiz Plugin parallel getestet und die Resultate verglichen, ob sie übereinstimmen.

3.6.5 Protokoll

Kopie der Tabelle aus der Systemtestspezifikation für die aktuelle Durchführung.

Use Case	implementiert	Fehler/Unschönheit	Status
UC1	Ja	Ist das Symbol „=“ in der Antwort, dann	Fehlerfrei

		wird der Text bis zum Symbol gekürzt.	
UC2	Ja	Gleiche Zuordnungen sind erlaubt, aber das System kann sie nicht zuordnen.	Fehlerfrei
UC3	Ja	Bei Zuordnungsfragen sind maximal 255 Zeichen (mit „M“ getestet) pro Zuordnung gestattet.	Fehlerfrei

3.7 Resultate

3.7.1 Zielerreichung

Moodle ist eine einfache Plattform zum Quiz erstellen. Mit dem Buch ist eine Zusammenfassung geschrieben worden und damit die 102 Quiz Fragen erfunden. Zu allen 12 Kategorien sind Fragen geschrieben, jede Kategorie hat jeweils eine Datei und zu jeder Kategorie ist mindestens eine Frage gestellt. Das Quiz sind GIFT-Format Dateien und mit dem Texteditor editierbar und sind im Klartext. Jeder Fragetyp hat eine eigene Syntax, die gut verständlich ist. Die Rechtschreibung ist mit Microsoft Word 2010 kontrolliert. Alle Quiz Dateien sind in Moodle importiert, durchgetestet und exportiert, auch mit QGis Quiz Plugin ist es durchgetestet und fehlerfrei.

3.7.2 Allgemeiner Erfahrungsbericht

Die Aufgabe war ein Datenbanksystem 1 Quiz, das mit Moodle genutzt werden kann. Da Moodle eine reine Onlineversion ist, wurde nach dem Projektstart beschlossen, dass das QGis Quiz Plugin als Offlinealternative schon bereits nutzbar ist und Fehler im Quiz Plugin direkt dem Entwickler gemeldet werden sollen, so dass jederzeit eine lauffähige Testumgebung da ist, um das Quiz zu erstellen. Moodle ist unvollständig dokumentiert und hat Fehler, zum Beispiel sind die Links von der Moodleplattform zur Moodle Dokumentation nicht verlinkt. Während dem parsen der GIFT-Format Dateien, meldet Moodle nur einfache Syntaxfehler, dabei hätte ich mir einiges an Zeit gespart mit einem besseren Parser.

3.7.3 Persönliche Erfahrungen

Pro Teammitglied ein Kapitel

3.7.3.1 David Tran

Im technischen Bericht (Teil 1).

3.8 Weiterentwicklung

Moodle hat weitere Fragetypen, zum Beispiel Wahr-Falsch-Fragen, Kurzantwort-Fragen, Numerische Fragen, welche die Vielfältigkeit der Fragensammlung verbessern. Das Quiz ist damit sicher abwechslungsreicher.

Oracle Datenbanken hat noch mehr Funktionen, die schwieriger sind, aber dafür weitaus mächtiger sind als PostgreSQL. Einen Haken könnte das GIFT-Format haben, das die SQL-Statements nicht richtig darstellen kann. Als Alternative wäre NoSQL.

4 Teil 2: Reengineering eines Desktop Konverters OGR

4.1 Überblick

Hier folgen die Kapitel gemäss Software-Engineering-Vorgehen.

4.2 Vision

Im technischen Bericht.

4.3 Anforderungsspezifikation

4.3.1 Allgemeine Beschreibung

4.3.1.1 Produkt Perspektive

Das Geoinformationssystem beschreibt wie und wo die Geodaten verarbeitet und gespeichert werden. OGR2GUI von Inventis ist eine Desktopapplikation, die Geoinformationen konvertiert. Als Quellen können Geodateien oder Ordner mit Geodateien, Datenbanken und Webservices sein. Die konvertierten Quellen werden an Zielen wie Dateien oder Ordner mit Dateien oder Datenbanken gespeichert.

4.3.1.2 Produkt Funktion

OGR2GUI ist eine grafische Benutzeroberfläche für die Konvertierung von GIS Datenformate. Mit einem Klick auf „Execute“ werden die Geodaten konvertiert. Im Ausgabefenster sind die Befehle aufgelistet, die an ogr2ogr übergeben werden und der gesamte Verlauf. Der Statusbalken zeigt den aktuellen Status an. Es benutzt die OGR Werkzeuge als Grundlage. Die Applikation konvertiert mit ogr2ogr, das ein Teil der Gdal Bibliothek ist.

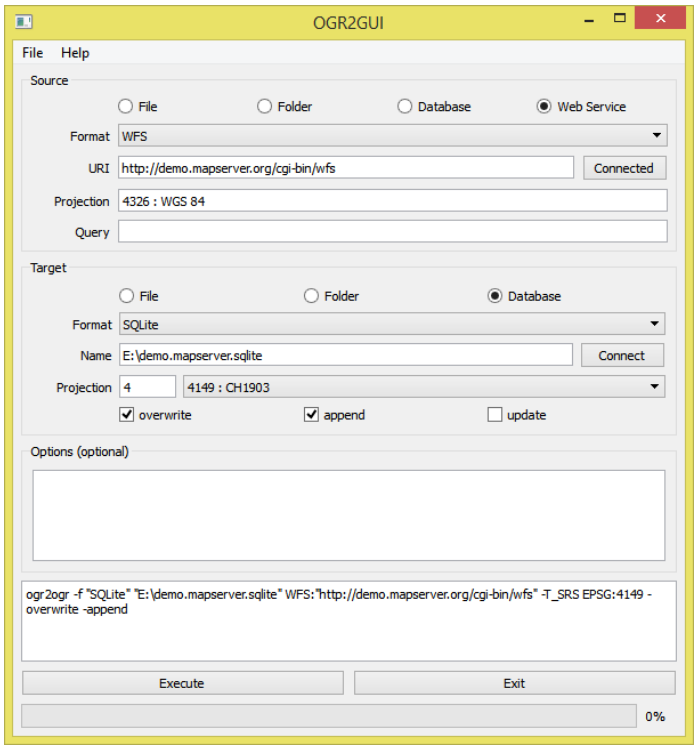


Abbildung 7 OGR2GUI

4.3.1.3 Benutzer Charakteristik

Die Endbenutzer, die nur wenige Computerkenntnisse haben oder die, welche die Kommandozeile selten benutzen.

4.3.1.4 Einschränkungen

OGR2GUI kann nur die vorgegebenen Quellen und Ziele konvertieren. Unter Webservice ist Web Feature Service implementiert, aber ogr2ogr schreibt damit nicht alle Zielformate. Ein Fehlerlogging zeigt folgende Formate an:

- Memory
- Interlis 1
- Interlis 2
- GPSBabel
- GFT
- GME

4.3.1.5 Annahmen

- Quelltexte sind nur so viel wie nötig erweitert.
- Die aktuellste Gdal (Geospatial Data Abstraction Library) Version ist 1.11.
- OGR2GUI binäre Datei ist mit shared Bibliotheken kompiliert.
- Nur für Windows Betriebssysteme ab Windows XP.
- Kompilierung mit Visual C++ 16.00 Compiler (x86 und x64).

4.3.1.6 Abhängigkeiten

Frameworks für die Entwicklung:

- Qt 5.2.1
- Qt Creator 3.1
- Microsoft Visual Studio 2010 Professional

4.3.2 Use Cases

Detailbeschreibung sämtlicher Use Cases.

4.3.2.1 Use Case Diagramm

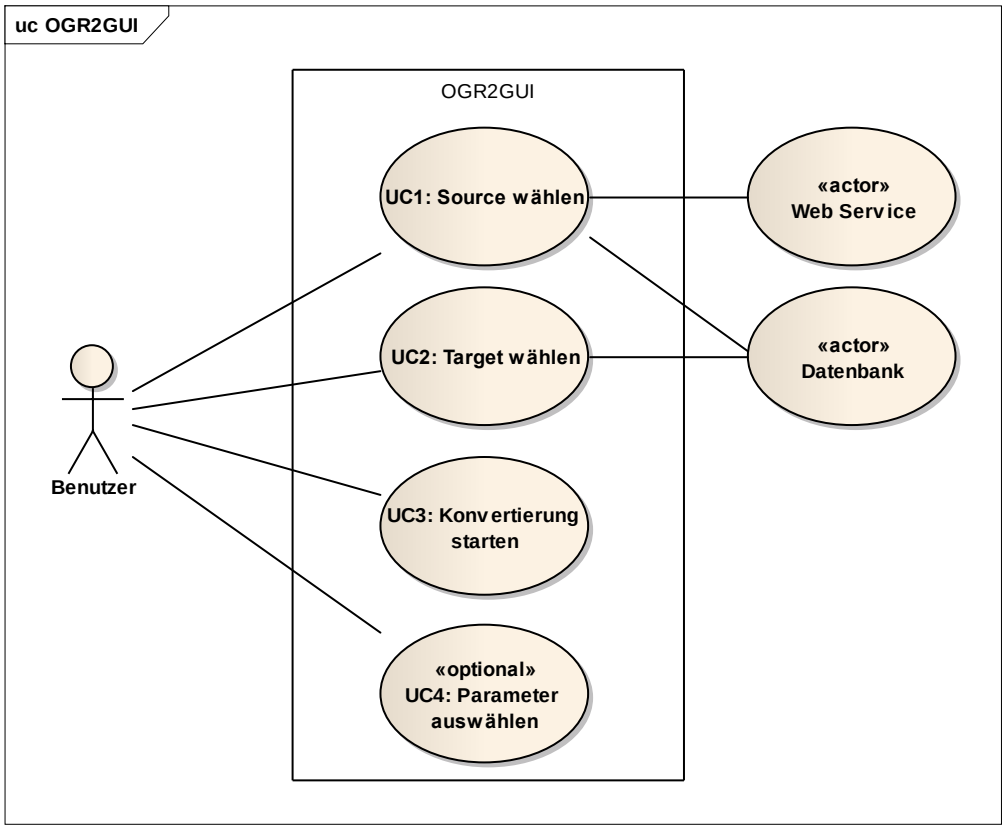


Abbildung 8 Use Case

4.3.2.2 Aktoren & Stakeholder

OGR2GUI als System:

- Benutzer:
  - Primary Actor
  - Ziel: Kann eine Quelle (Source)/Ziel (Target) und ein Geoformat auswählen. Die Konvertierung starten.

4.3.2.3 Beschreibungen (Brief)

Alle Use Cases in einzelnen Kapiteln im Brief Format beschrieben.

4.3.2.3.1 Source auswählen

Der Benutzer kann eine Quelle (Source) auswählen. Zur Auswahl ist Datei (File), Ordner (Folder), Datenbank (Database), Webservice. Nach dem ein Ziel gewählt wurde, wird die Formateliste aktualisiert. Der Benutzer sucht das Geoformat aus der Formateliste.

4.3.2.3.2 Target auswählen

Der Benutzer kann ein Ziel (Target) auswählen. Zur Auswahl ist Datei (File), Ordner (Folder), Datenbank (Database). Nach dem ein Ziel gewählt wurde, wird die Formateliste aktualisiert. Die Geoformate sind als Liste dargestellt. Der Benutzer sucht das Geoformat aus der Formateliste. Eine Projektionssuche kann über die Suchfunktion nach einer Projektion suchen. Die Eingabe des Suchbegriffs werden die Projektionen mit diesem Ausdruck gefiltert und angezeigt. Gibt es die Projektion nicht, bleibt die Anzeige bei der letzten Projektion, die gefunden wurde oder die Anzeige bleibt leer.

4.3.2.3.3 Konvertierung starten

Die Konvertierung muss der Benutzer starten. Dieser Schritt kann nach der ersten Konfiguration beliebig oft wiederholt werden.

4.3.2.4 Beschreibungen (Fully Dressed)

Spezielle und wichtige Use Cases in einzelnen Kapiteln im Fully Dressed Format beschrieben.

4.3.2.4.1 Source wählen

Scope	OGR2GUI
Level	user-goal
Primary Actor	Benutzer
Stakeholder and Interests	Möchte eine Quelle (Source) konvertieren.
Preconditions	Die Quelle ist implementiert.
Success Guarantee	Auswahl aus der Formateliste.
Main Success Scenario	1. Der Benutzer wählt eine Quelle aus. 2. Der Benutzer wählt ein Format aus der Formateliste aus. 3. Die Quelle kann geöffnet/verbunden werden.
Extensions	1. Das Quellformat ist falsch. a. Ein Format aus der Formateliste auswählen. 2. Die Quelle kann keine Datenbankverbindung aufbauen.
Special Requirements	Kenntnisse in Geoinformationssysteme.
Technology and Data Variations List	Keine.
Frequency of Occurrence	Jedes Mal wenn eine Quelle gewählt wird.
Miscellaneous	Keine.

4.3.2.4.2 Target wählen

Scope	OGR2GUI
Level	user-goal
Primary Actor	Benutzer
Stakeholder and Interests	Möchte ein Ziel (Target) vorgeben.
Preconditions	Das Ziel ist implementiert.
Success Guarantee	Auswahl aus der Formateliste.
Main Success Scenario	1. Der Benutzer wählt ein Ziel aus. 2. Der Benutzer wählt ein Format aus der Formateliste aus. 3. Das Ziel kann gespeichert/abgelegt werden.
Extensions	1. Das Ziel hat keine Schreibrechte. a. Speicherort ändern. 2. Das Ziel kann keine Datenbankverbindung aufbauen.
Special Requirements	Kenntnisse in Geoinformationssysteme.
Technology and Data Variations List	Keine.
Frequency of Occurrence	Jedes Mal wenn ein Ziel gewählt wird.
Miscellaneous	Keine.

4.3.2.4.3 Konvertierung starten

Scope	OGR2GUI

Level	user-goal
Primary Actor	Benutzer
Stakeholder and Interests	Möchte eine Konvertierung starten.
Preconditions	UC1 und UC2 sind bekannt.
Success Guarantee	Quelle und Ziel sind ausgewählt.
Main Success Scenario	1. Die Konvertierung startet. 2. Die Konvertierung ist erfolgreich.
Extensions	1. Die Konvertierung startet nicht. a. UC1 und/oder UC2 Parameter neu konfigurieren. 2. Die Konvertierung ist nicht erfolgreich. a. Konvertierung 1x wiederholen.
Special Requirements	Kenntnisse in Geoinformationssysteme.
Technology and Data Variations List	Keine.
Frequency of Occurrence	Jedes Mal wenn eine Konvertierung startet.
Miscellaneous	Keine.

4.3.3 Weitere Anforderungen

4.3.3.1 Qualitätsmerkmale

Beschreibung der Qualitätsmerkmale der Software (ISO 9126).

4.3.3.1.1 Funktionalität

Implementiert sind öffnen und speichern von Dateien, Verbindungen zu Datenbanken und Webservice anhand eines Beispiels mit Web Feature Service (WFS) und nur schreibgeschützt.

4.3.3.1.1.1 Angemessenheit

Datenbanksysteme sind für die Datenbankverbindungen vorausgesetzt.

4.3.3.1.1.2 Richtigkeit

WFS baut eine Verbindung mit einem Geoinformationssystem (GIS) auf und ruft Geodaten vom Server ab.

4.3.3.1.2 Benutzerbarkeit

Datenbankverbindungen muss der Benutzer konfigurieren. Die Auswahl „WFS“ bietet an, eine Uniform Resource Identifier (URI) einzugeben.

4.3.3.1.2.1 Bedienbarkeit

Alle Datenbanksystemkonfigurationsparameter müssen bekannt sein für den Verbindungsaufbau zum Server:

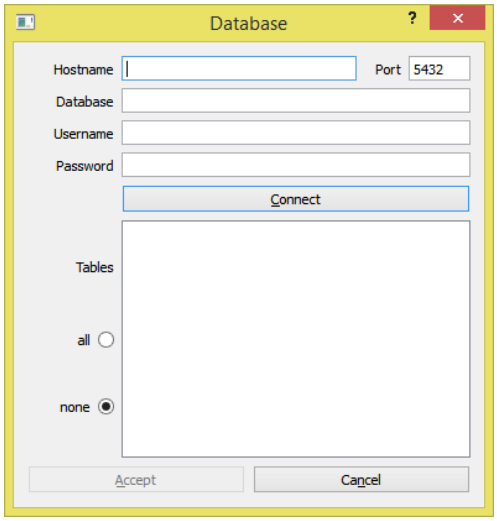


Abbildung 9 OGR2GUI Fenster

4.3.3.1.2.2 Attraktivität

Die grafische Oberfläche zeigt bei Webservice mit einem Hinweis auf dem Button, während der Benutzer die URI eingibt, an, ob die Verbindung zum Server erfolgt ist.

4.3.3.1.3 Übertragbarkeit

Die Software ist mit VC10 kompiliert und ist für Microsoft Betriebssysteme ab Windows XP.

4.3.3.1.3.1 Installierbarkeit

Die Software und die Abhängigkeiten müssen in dasselbe Installationsverzeichnis.

4.3.3.2 Schnittstellen

4.3.3.2.1 Geodaten

„Geodaten: raumbezogene Daten, die mit einem bestimmten Zeitbezug die Ausdehnung und Eigenschaften bestimmter Räume und Objekte beschreiben, insbesondere deren Lage, Beschaffenheit, Nutzung und Rechtsverhältnisse;“[2]

4.3.3.2.2 Datenbank

Die Datenbank ist ein System, die grosse Mengen an Daten speichert. Die Aufgabe ist es die Datenmengen für unterschiedliche Zwecke bereitzustellen.

4.3.3.2.3 Webservice

Es ist eine Software, die eine Verbindung mittels URI zu einem Server aufstellt. Der Server antwortet dann mit den geforderten Informationen.

4.3.3.3 Randbedingungen

4.3.3.3.1 Geospatial Data Abstraction Library (GDAL)

Eine Bibliothek mit Funktionen für Rasterdaten. OGR ist in dieser Bibliothek enthalten und hat Funktionen für Vektordaten.

4.3.3.3.2 Web Feature Service (WFS)

Damit kann Geodaten von einem GIS heruntergeladen werden.

[2] Art. 3 Abs. 1 Bst. a GeoIG (SR 510.62)

4.4 Analyse

4.4.1 Domain Modell

4.4.1.1 Strukturdiagramm

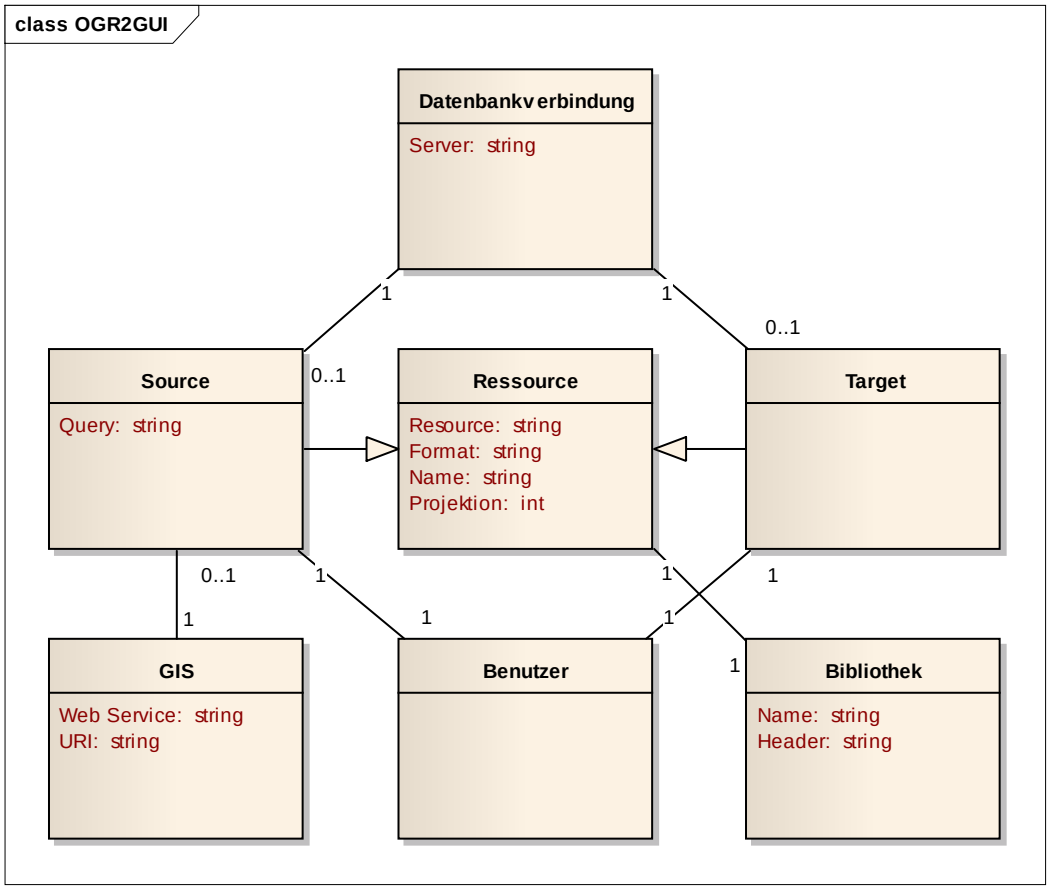


Abbildung 10 Domain Modell

4.4.1.2 Wichtige Konzepte

4.4.1.2.1 Source
Die Quelle öffnet Ressourcen von Geodaten, Datenbanken oder GIS.

4.4.1.2.2 Target
Das Ziel speichert Ressourcen als Geodaten oder in eine Datenbank.

4.4.1.2.3 Datenbankverbindung
Der Server hat eine Kennung. Die Datenbankverbindung nutzt diese Kennung.

4.4.1.2.4 GIS
Webservice ist ein GIS. Es hat ein URI als Kennung.

4.4.1.2.5 Bibliothek
Die Bibliothek ist für die Konvertierung. Die Ressource ändert dabei das Verhaltensmodell, aber mit denselben Informationen.

4.4.2 Systemsequenzdiagramme

4.4.2.1 UC1: Source wählen

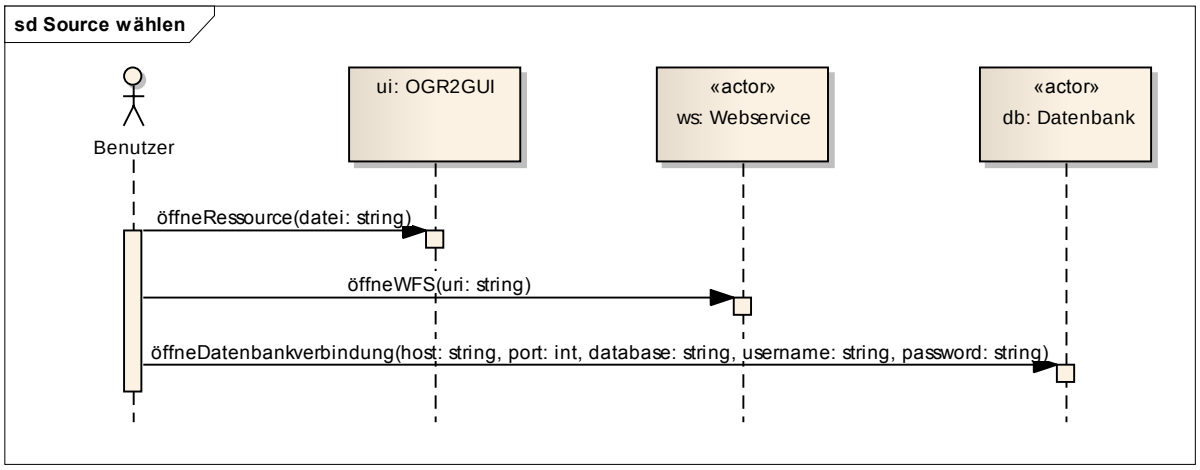


Abbildung 11

4.4.2.2 UC2: Target wählen

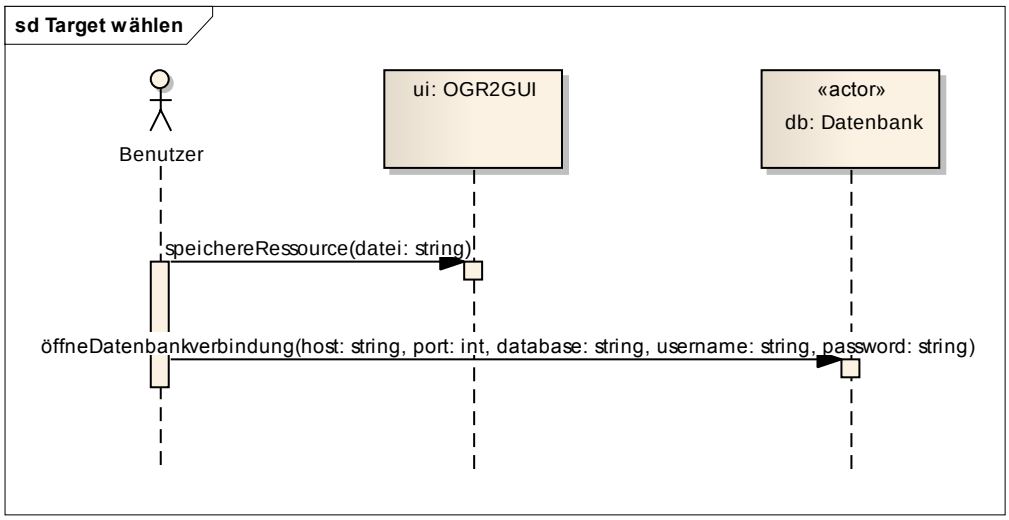


Abbildung 12

4.4.2.3 UC3: Konvertierung starten

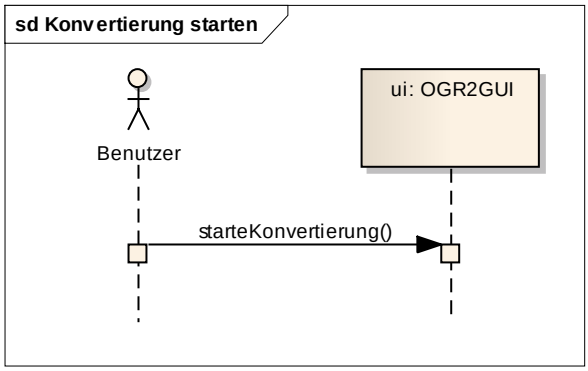


Abbildung 13

4.4.3 Systemoperationen

4.4.3.1 Vertrag Systemoperation öffneRessource()

Beschreibung	Fügt eine Ressource hinzu.
Preconditions	Die Ressource existiert.
Postconditions	Die Ressource ist geöffnet.

4.4.3.2 Vertrag Systemoperation öffneWFS()

Beschreibung	Öffnet eine Verbindung zum Webservice WFS mit der URI.
Preconditions	- Das GIS existiert. - Die WFS URI ist bekannt.
Postconditions	Die Ressourcen vom WFS sind abgerufen.

4.4.3.3 Vertrag Systemoperation öffneDatenbankverbindung()

Beschreibung	Öffnet eine Datenbankverbindung zum Server.
Preconditions	- Der Server ist existiert. - Die Datenbankkonfiguration ist bekannt. - Geodaten sind in den Datenbanktabellen eingefügt.
Postconditions	Die Verbindung zur Datenbank ist aufgebaut.

4.4.3.4 Vertrag Systemoperation speichereRessource()

Beschreibung	Legt eine Ressource an.
Preconditions	- Die Ressourcenattribute sind gesetzt. - Die Ressource hat eine Schreibposition.
Postconditions	Die Ressource ist gespeichert.

4.4.3.5 Vertrag Systemoperation starteKonvertierung()

Beschreibung	Die Konvertierung schreibt eine Ressource neu nach den vorgegebenen Parametern.
Preconditions	- Die Parameter für die neue Ressource sind gesetzt. - Die Ressource kann abgelegt werden.
Postconditions	- Die Ressource wurde gespeichert. - Die Ressource existiert.

4.5 Design (Entwurf)

4.5.1 Systemübersicht

Beschreibt die Softwarearchitektur eines Systems und wie sie sich präsentiert.

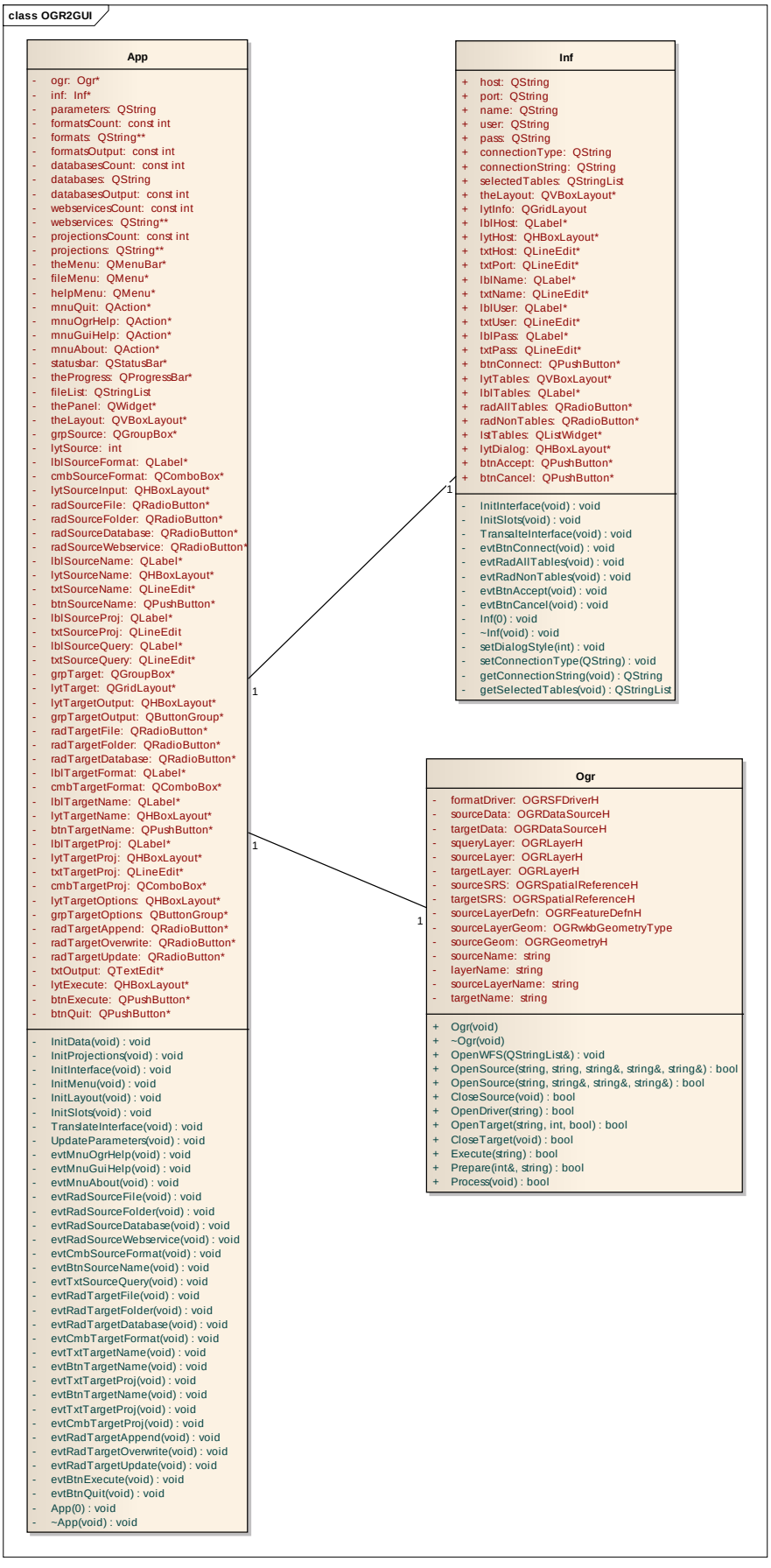


Abbildung 14

4.5.2 Architektonische Ziele & Einschränkungen

4.5.2.1 Distribution

- OGR2GUI 0.7
- GDAL 1.11

4.5.2.2 Implementationsstrategie

- Framework aktualisieren
- Web Feature Service implementieren
- Formate ergänzen
- Optioneneingabe hinzufügen

4.5.2.3 Design

- Model View Controller (MVC)
- Datenbankverbindung mit SQLite

4.5.2.4 Entwicklungstools

- Qt 5.2.1
- Qt Creator 3.1
- Mingw-builds
- Visual C++ 16.00 (x86, x64)
- Doxygen 1.8.7

4.5.3 Logische Architektur

Beschreibung der logischen Struktur des Projekts.

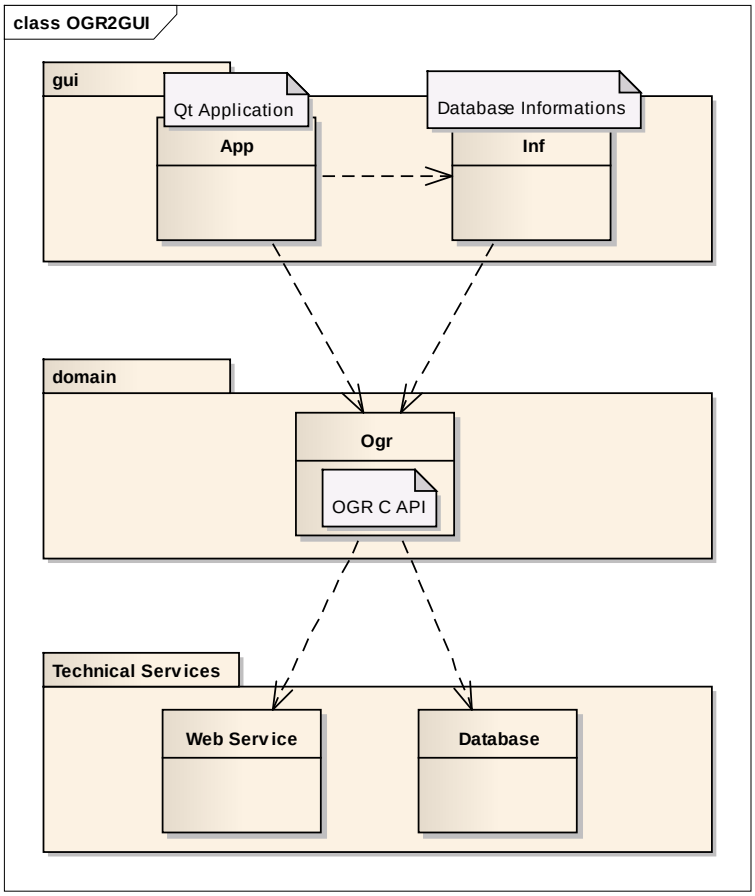


Abbildung 15

4.5.3.1 Gui/App

4.5.3.1.1 Klassenstruktur

Klasse	**Beschreibung**
InitData(void)	Initialisiert alle Formate, Datenbanken, Projektionen und Webservices aus der Datei Dta.h.
InitInterface(void)	Die graphische Benutzeroberfläche initialisieren.
UpdateParameters(void)	Das Logfenster aktualisieren.
evtTxtSourceName(void)	Nach jeder Eingabe in Source->Name: - Bei Webservice überprüft es die URI - Ansonsten schaut es nach, ob es Projektionen hat.
evtBtnSourceName(void)	Geodaten einlesen. Datenbankverbindung herstellen.
evtBtnTargetName(void)	Geodaten schreiben.
evtBtnExecute(void)	Geodaten aus der Ressource laden und konvertieren.

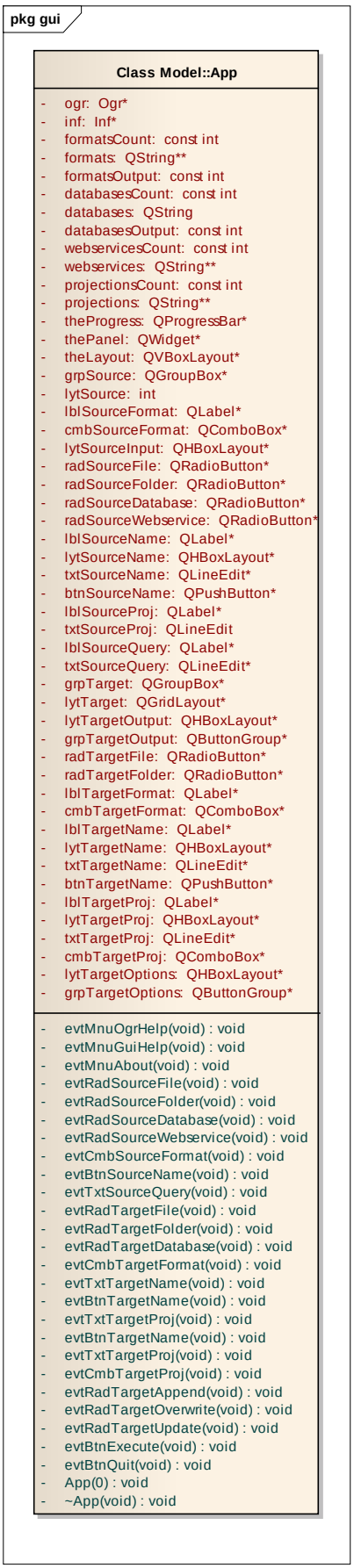


Abbildung 16

4.5.3.1.2 Schnittstellen

Die Schnittstelle ist die Qt API, die in der Klasse App implementiert ist.

4.5.3.2 Domain/Ogr

4.5.3.2.1 Klassenstruktur

Klasse	**Beschreibung**
OpenWFS(QStringList)	Öffnet die URI.
OpenSource(filename, epsg, query, error) OpenSource(filename, layername, epsg, query, error)	Öffnet die Quellressource.
OpenDriver(driver)	Öffnet den Treiber für das Format.
OpenTarget(filename, projection, update)	Öffnet die Zielressource.
Prepare(features, query)	Wartet auf die Ressource.
Process(void)	Konvertiert die Ressourcen.
CloseTarget(void)	Schliesst das geöffnete Target.
CloseSource(void)	Schliesst die geöffnete Source.

Abbildung 17

4.5.3.2.2 Schnittstellen

Die Schnittstelle ist die OGR API, die in der Klasse Ogr implementiert ist.

4.5.3.2.3 Wichtige interne Abläufe

Klasse	**Beschreibung**
OGRRegisterAll()	Alle Formate registrieren.
OGROpen(filename, rw, driver)	Öffnet eine Ressource.
OGR_DS_GetLayerByName(sourceHandler, layername)	Lädt einen Layer anhand vom Namen des Layers.
OGR_L_GetNextFeature(layer)	Lädt das nächste Feature vom Layer.
OGR_L_GetLayerDefn(layer)	Lädt das Schema des Layers.
OGR_DS_Destroy(datasource)	Schliesst die geöffnete Ressource.
OGR_DS_ExecuteSQL(datasource, query,	SQL Statement ausführen.

geom, dialect)	
OGR_Dr_CreateDataSource(driver, name, option)	Erstellt eine Ressource anhand des Treibers.
OGR_DS_CreateLayer(handle, layername, handle, geom, options)	Erstellt ein Layer.

4.5.3.3 Technical Services/Inf

4.5.3.3.1 Klassenstruktur

Klasse	**Beschreibung**
InitInterface(void)	Initialisiert die grafische Benutzeroberfläche.
evtBtnConnect(void)	Datenbankverbindungsaufbau.
evtBtnAccept(void)	Lädt die Tabellen aus der Datenbank.

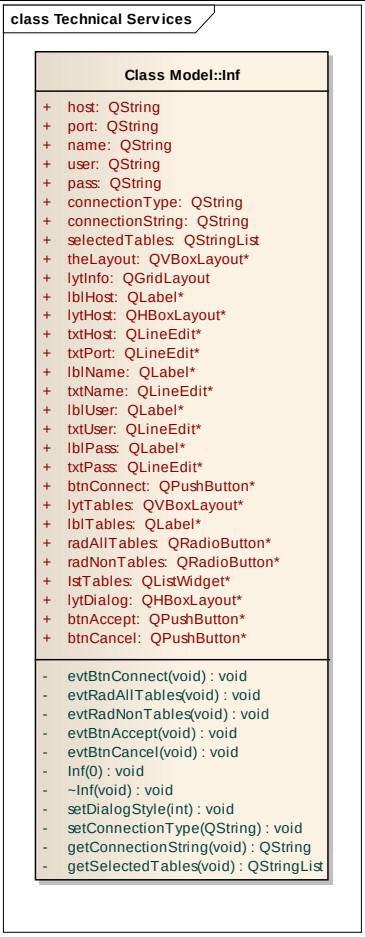


Abbildung 18

4.5.3.3.2 Schnittstellen

Die Schnittstelle ist die Datenbankinitialisierung, die in der Klasse Inf implementiert ist.

4.5.3.4 Technical Services/Web Service

Klasse	**Beschreibung**
InitInterface(void)	Initialisiert die grafische Benutzeroberfläche.
evtBtnConnect(void)	Verbindungsaufbau mit einem GIS.
evtBtnAccept(void)	Lädt die Layers vom GIS

4.5.3.4.1 Schnittstellen

Die Schnittstelle ist die WFS-Verbindung, die in der Klasse wfsConnect implementiert ist.

4.5.3.5 Wichtige Abläufe

Die Dta.h Datei hat alle Formate für Dateien, die unterstützten Datenbanken und Webservices gespeichert.

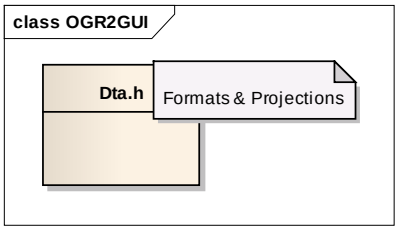


Abbildung 19 Ohne Projections

4.5.4 Prozesse und Threads

Die Threads und Prozesse Implementationen sind mit einem Macro nach den Architekturen x86/x64/Wow64 abgefragt.

OGR2GUI x64 auf einer Windows x64 Plattform:

Die Klasse ogr2ogrThread implementiert den QThread. In diesem Thread startet der Prozessaufruf, der ogr2ogr aufruft. Einen Stream, die alle Ausgaben von ogr2ogr aufnimmt, schreibt der Prozess in eine Logdatei. Der Prozess wartet und beendet erst wenn nichts blockiert.

OGR2GUI x86 auf einer Windows x64 Plattform (Wow64):

Der Prozess QProcess startet ohne Thread (ogr2ogrThread) direkt in der Klasse Ogr.cpp, weil der QProcess im QThread nicht startet.

OGR2GUI x86 auf einer Windows x86 Plattform:

Der Prozess ist eine Implementation mit der Windows API Funktion *CreateProcess*, weil QProcess gar nichts aufruft.

4.5.5 Deployment

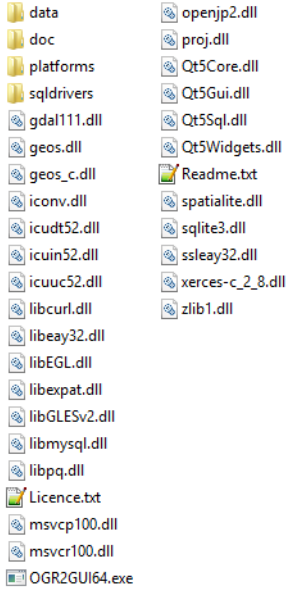


Abbildung 20

Voraussetzung ist ein Betriebssystem ab Windows XP. Im Ordner data sind die Projektionsdateien. Die binäre Datei ogr2gui.exe startet das Programm OGR2GUI.

4.5.6 Datenspeicherung

OGR2GUI kann mit den Datenbanksysteme umgehen. Für eine Datenbankverbindung müssen Datenbankserver installiert sein:

- PostgreSQL
- MySQL
- ODBC

Nur SQLite muss keine Datenbankverbindung herstellen, weil die Datenbank aus einer Datei besteht.

4.6 Implementation (Entwicklung)

4.6.1 Die Qt Creator Konfigurationsdatei

```
TEMPLATE = app
INCLUDEPATH += $$PWD/include $$PWD/include/ogr

HEADERS += \
   include/App.h \
   include/Dta.h \
   include/Ogr.h \
   include/Inf.h \
   include/utils.h \
   include/ogr2ogrThread.h \
   include/wfsConnect.h

SOURCES += \
   src/Ogr.cpp \
   src/Inf.cpp \
   src/App.cpp \
   src/main.cpp \
   src/utils/ogr2ogr.cpp \
   src/utils/commonutils.cpp \
   src/ogr2ogrThread.cpp \
   src/wfsConnect.cpp

CONFIG += c++11
QT += sql widgets

win32: contains(QMAKE_TARGET.arch, x86) {
            TARGET = ogr2gui
            LIBS += -L$$PWD/lib -lgdal_i
            DESTDIR += $$PWD/bin
}

win32: contains(QMAKE_TARGET.arch, x86_64) {
            TARGET = ogr2gui64
            LIBS += -L$$PWD/lib -lgdal_i_x64
            DESTDIR += $$PWD/bin
}
```

Erklärung:

Option	Beschreibung
TEMPLATE = app	qmake kompiliert eine Applikation
INCLUDEPATH	Includepfad aller include Dateien
HEADERS	Alle Header Dateien auflisten
SOURCES	Alle Source Dateien auflisten
CONFIG	Konfigurationsoptionen (hier: c++11)
QT	Zusätzliche Qt Bibliotheken auflisten
TARGET	Applikationsname
LIBS	Individuelle Bibliotheken (hier: gdal_i)
DESTDIR	Zielordner
QMAKE_TARGET.arch	Zielarchitektur (x86, x86_64)

Die ogr2gui.pro Datei hat Konfigurationsoptionen, die für die Kompilierung verwendet werden. Obligatorisch sind TEMPLATE, INCLUDEPATH HEADERS, SOURCES, QT,

TARGET und LIBS damit die Konfigurationen vollständig sind. Die restlichen Optionen wie zum Beispiel CONFIG, QMAKE_TARGET.arch und DESTDIR spezifizieren nur genauer wie es kompiliert werden soll.

4.6.2 Beschreibungen von Klassen und Funktionen

```
#pragma comment(linker, "/SUBSYSTEM:CONSOLE")
#include "App.h"
#include "cpl_conv.h"
#include <iostream>

int main(int argc, char **argv) {
    QApplication app(argc, argv);
    string dataPath = QDir::toNativeSeparators(QCoreApplication::applicationDirPath() +
QDir::separator() + "data").toStdString();
```

Die Zeile ruft den absoluten Pfad auf, in welcher die binäre Datei liegt. Der Ordner „data“ wird zum absoluten Pfad hinzugefügt. Die folgende *CPLSetConfigOption* setzt diesen absoluten Pfad mit dem Ordner „data“ als Umgebungsvariable. Im Ordner data sind alle Projektionsdateien, die von GDAL und OGR2GUI unterstützt sind.

```
    CPLSetConfigOption("GDAL_DATA", dataPath.c_str());
    if(1 < argc) {
        for(int i=0;i<argc;++i) {
            std::cout << argv[i] << " ";
        }
        std::cout << std::endl;
        return ogr2ogr(argc, argv);
    } else {
        new App();
    }
    return app.exec();
}
```

Wenn mehr als ein Argument der Main-Funktion übergeben wurde, dann startet ogr2gui anstatt die grafische Benutzeroberfläche, das Konsolentool ogr2ogr. Alle bekannten ogr2ogr Optionen können verwendet werden.

```
void App::UpdateParameters(void) {
    parameters = tr("ogr2ogr");
    parameters += currentParameters();
    if(radSourceWebservice->isChecked())
        parameters += tr(" ") + wfs->getSelectedLayers();
    if(!txtInput->toPlainText().isEmpty())
        parameters += tr(" ") + txtInput->toPlainText().simplified();
    txtOutput->setText(parameters);
}
```

```
QString App::currentParameters() {
   QString parameters = tr(" -f ") + tr("\"") + cmbTargetFormat->currentText() + tr("\" ");
   parameters += tr("\"") + txtTargetName->text()+ tr("\" ");
   if(radSourceWebservice->isChecked() && !cmbSourceFormat->currentText().isEmpty())
      parameters += webservices[cmbSourceFormat->currentIndex()][1];
   parameters += tr("\"") + txtSourceName->text() + tr("\"");
   if(!cmbTargetProj->currentText().isEmpty()) {
      parameters += tr(" ") + tr("-T_SRS");
      parameters += tr(" EPSG:") + projectionsList.at(cmbTargetProj->currentIndex()).first;
   }
   if(!txtSourceQuery->text().isEmpty())
      parameters += tr(" -sql ") + tr("\"") + txtSourceQuery->text() + tr("\"");
   if(radTargetOverwrite->isChecked())
      parameters += tr(" -overwrite");
   if(radTargetAppend->isChecked())
      parameters += tr(" -append");
   if(radTargetUpdate->isChecked())
      parameters += tr(" -update");
   return parameters;
}
```

Damit die Parameter in der richtigen Sortierung weitergegeben werden können, sind diese in einem QString abgespeichert. In der grafischen Benutzeroberfläche sind folgende Parameter implementiert:

ogr2ogr -f „*Zielformat*“ „*Ziel*“ „*[WFS:]Quelle*“ -T_SRS *EPSG:[1234]* -sql „*SELECT * from [xyz]*“ *-overwrite* [*-append*, *-update*] *layer1 layer2 layer3 …*

```
void App::evtBtnExecute( void )
{
   UpdateParameters();

   QString sourcename = txtSourceName->text();
   QString targetname = txtTargetName->text();
   string epsg;
   string query;
   string error;

   bool resVal = true;
   if(radSourceWebservice->isChecked()) {
      QStringList fileList = wfs->getSelectedLayersAsList();
      sourcename = webservices[cmbSourceFormat->currentIndex()][1] + sourcename;
      for(int i=0;i<fileList.size();++i) {
         if(!ogr->OpenSource(sourcename.toStdString(), fileList.at(i).toStdString(), epsg,
query, error)) {
            resVal = false;
            return;
         }
      }
   } else {
      resVal = ogr->OpenSource(sourcename.toStdString(), epsg, query, error);
   }
```

Mit Hilfe der OGR API überprüft die Funktion *ogr->OpenSource(string filename, string &epsg, string &query, string &error)*, ob die Quelle geöffnet werden kann. Ist die Quelle das Web

Feature Service, dann öffnet die Funktion jeden Layer genau einmal. Bei einem Fehler zeigt OGR2GUI eine Fehlermeldung im Logfenster.

```
  if(resVal) {
     if(ogr->OpenDriver(cmbTargetFormat->currentText().toStdString())) {
        ogr->TestProjection((projectionsList.at(cmbTargetProj->currentIndex()).first).toInt());
        if(!radSourceDatabase->isChecked())
           ogr->TestFeature();
        ogr->CloseSource();
```

Die Funktion *ogr->OpenDriver(string drivername)* schaut nach, ob der Treiber in der GDAL Bibliothek existiert und zeigt eine Fehlermeldung im Logfenster im anderen Fall, wenn der Treiber nicht gefunden wurde. Die Funktionen *ogr->TestProjection(int projection)* testet die Projektion und *ogr->TestFeature(void)* testet die Feature noch vor der Konvertierung auf Fehler.

```
        theProgress->setValue(0);
        QString parameters = currentParameters();
        QString path = QDir::toNativeSeparators(QCoreApplication::applicationFilePath());
        path += parameters;
        if(!wfs->getSelectedLayers().isEmpty()) {
           QStringList wfsLayerList = wfs->getSelectedLayersAsList();
           for(int i=0;i<wfsLayerList.size();++i) {
              QString layer = tr(" ") + wfsLayerList.at(i).simplified();
              QString pathTemp = path;
              pathTemp += layer;
              txtOutput->append(sourcename + tr(" ") + layer + tr(" > ") + targetname + tr(" ...
"));
              if(!ogr->OpenOgr2ogr(pathTemp, btnExecute))
                 txtOutput->append(tr("\n * unable to open ogr2ogr !\n"));
              int progressValue = i*100/wfsLayerList.size();
              if(progressValue <= 100)
                 theProgress->setValue(progressValue);
           }
```

Die if-Anweisung ruft die WFS Layers ab und wenn Layers vorhanden sind, dann werden diese Layer per Layer an die ogr2ogr übergeben. Der Statusbalken in der grafischen Oberfläche zeigt den Fortschritt an, der bei jeder Iteration bis höchstens 100 erhöht wird.

```
        } else {
            theProgress->setMinimum(0);
            theProgress->setMaximum(0);
            txtOutput->append(sourcename + tr(" > ") + targetname + tr(" ... "));
            if(!txtInput->toPlainText().isEmpty()) {
                path += tr(" ") + txtInput->toPlainText();
            }
            if(!ogr->OpenOgr2ogr(path, btnExecute))
                txtOutput->append(tr("\n * unable to open ogr2ogr !\n"));
        }
        theProgress->setValue(100);
        theProgress->setMaximum(100);
    } else {
        txtOutput->append(tr("\n * unable to open driver !\n"));
    }
} else {
    txtOutput->append(tr("\n * unable to open source !\n"));
}
}
```

Sind keine WFS Layers angegeben, dann zeigt der Statusbalken während der Konvertierung nur eine Warteschleife.

```
bool Ogr::OpenWFS(QString uri, QStringList &fileList) {
    sourceName = uri.toStdString();
    OGRDataSourceH sourceData = OGROpen(sourceName.c_str(), 0, NULL);
    if(sourceData != NULL) {
        for(int i = 0; i < OGR_DS_GetLayerCount(sourceData); ++i) {
            OGRLayerH sourceLayer = OGR_DS_GetLayer(sourceData, i);
            if(sourceLayer != NULL) {
                OGRFeatureDefnH sourceLayerDefn = OGR_L_GetLayerDefn(sourceLayer);
                fileList.append(OGR_FD_GetName(sourceLayerDefn));
            }
        }
        return true;
    }
    return false;
}
```

Diese Funktion öffnet ein URI und legt die Layernamen in ein QStringList ab. Die Layernamen werden für die Konvertierung mit ogr2ogr verwendet. Die grafische Benutzeroberfläche zeigt eine Layerauswahl anhand dieses QStringList an.

4.7 Test

4.7.1 Voraussetzungen

- OGR2GUI (Quelltexte und Abhängigkeiten)
- Qt Creator 3.1
- Microsoft Visual Studio 2010 Professional

4.7.2 Vorbereitungen

Die Datei ogr2gui_test.pro mit Qt Creator 3.1 öffnen und einen Buildordner auswählen. Im Ordner „bin“ sind die Ordner „data“ und „test“, die in den Buildordner müssen. Im Verzeichnis „msvc-x86_dll“ sind Dateien und Ordner. Alles ebenfalls in den Buildordner kopieren. Die Datei „ogr2gui_test.exe“ ausführen. Die Qt-Tests müssen alle bestanden sein.

4.7.3 Systemtest

Beschreibung der einzelnen Tests der Use Cases.

Use Case	Beschreibung
UC1	1. Eine Ressource/WFS/Datenbankverbindung öffnen. 2. Ein Format auswählen. 3. (Eine neue SQL Query erstellen.)
UC2	1. Eine Ressource/Datenbankverbindung öffnen. 2. Ein Format auswählen. 3. (Falls Projektionen vorhanden sind, darf eine neue Projektion ausgewählt werden). 4. Default: overwrite (append, update).
UC3	1. Die Konvertierung mit „Execute“ starten.

4.7.4 Angaben zur Durchführung

Der Systemtest ist mit der Version OGR2GUI 0.7 32-bit durchgeführt. Es ist mit Qt Creator 3.1 und VC10 Pro 32-bit kompiliert. Die Testumgebung ist Windows XP 32-bit, Windows 7 64-bit und Windows 8.1 64-bit. Die Testdateien sind im Ordner „test“. Der Qt-Test mit „ogr2gui_test.exe“ ist bereits mindestens einmal auf jedem Betriebssystem ausgeführt und es sind alle bestanden. Die grafische Benutzeroberfläche kann man nur manuell testen, in dem die bereitgestellten Testdateien im Ordner „test“ mit OGR2GUI konvertiert. Mit QGis kompatiblen Daten sind alle konvertierten Dateien mit den Quellressourcen angeschaut und visuell verglichen. Auch mit dem Kommandozeilentool ogr2ogr nochmals konvertiert und die Dateien nochmals mit QGis verglichen. QGis bietet an, die Tabellen anzuschauen und somit auch den Inhalt derselben Dateien parallel anzuschauen. Einige Formate sind im Textformat, so dass diese als ASCII mit einem Difftool wie zum Beispiel Beyond Compare überprüft sind.

4.7.5 Protokoll

Kopie der Tabelle aus der Systemtestspezifikation für die aktuelle Durchführung.

Use Case	implementiert	Fehler/Unschönheit	Status
UC1	Ja	Projektionen sind geparst, aber ohne den entsprechenden Eintrag (EPSG) im gcs.csv, zeigt es nicht an.	Fehlerfrei
UC2	Ja	Ohne gcs.csv Datei von GDAL im Ordner „data“, stehen keine Projektionen zur Auswahl. Die grafische Benutzeroberfläche zeigt beim Start eine Fehlermeldung.	Fehlerfrei
UC3	Ja	Während der Konvertierung wechselt der Statusbalken sofort zu 100%, sobald alle Ressourcen an ogr2ogr gegeben wurden. Der Button „Execute“ dagegen bleibt deaktiviert bis alle Thread gestartet sind.	Fehlerfrei

4.7.6 Verbesserungsmöglichkeiten

4.7.6.1 Bekannte Einschränkungen
Web Feature Service lädt nur im 64-bit Modus.

4.7.6.2 Mögliche Detailverbesserungen
Web Feature Service im 32-bit Modus vollständig benutzen können.

4.8 Resultate

4.8.1 Zielerreichung
OGR2GUI 0.7 ist mit Qt 5.2.1 aktualisiert. Es sind vor allem visuelle Veränderungen im Vergleich mit Qt 4.5.2. Speziell zu erwähnen sind die visuellen Anpassungen an die neuen Betriebssysteme wie „Look and Feel“. Nur wenig Quelltext musste man ändern um aus OGR2GUI eine Qt 5 Applikation zu machen. OGR2GUI kompiliert mit MinGW 32-bit, Mingw-builds und Microsoft Visual Studio ab 2010. Die Treiber für GDAL 1.11 waren dafür wählerischer als es um die Version des VC ging, so dass VC10 genommen wurde. Die wichtigsten Treiber sind damit in GDAL. Diese Bibliothek ist in OGR2GUI 0.7. Web Feature Service ist implementiert. Bei dieser Auswahl zeigt es ein Dialogfenster an und mit einer URI und dem Button „connect“ wird eine Verbindung mit dem GIS hergestellt. Auf dem Client werden alle Informationen verarbeitet und dann die Layerauswahl aktualisiert. Ein neues Optionenfenster oberhalb des Logfensters ist neu dazugekommen. Ins Optionenfenster können alle Optionen von ogr2ogr zusätzlich hinzugefügt werden und ist optional.

4.8.2 Allgemeiner Erfahrungsbericht
Die Aufgabe im Teil 2 war Ogr2gui 0.6 wieder für eine Softwareverteilung bereitzustellen und es soll wieder mit denselben Funktionalitäten sein. Die letzte Version hat immer noch sehr viele Downloads und die Webpräsenz ist laut den Entwicklern sehr gut besucht. Angefangen wurde mit dem Compiler Mingw und es wurde verworfen, weil Datenbanken wie PostgreSQL und MySQL damit nicht kompilierten und daraufhin beschlossen eine GDAL mit VC10 zu nehmen. Schade war es vor allem, weil es verschiedene Mingw x64 Builds gibt und keine davon zum „Ur“-Mingw kompatibel ist. Die GDAL Dokumentation ist nur eine API Dokumentation. Die OGR API ist für Checks vor dem Aufruf von ogr2ogr und Qt-Test verwendet. Die Qt-Tests öffnen und vergleichen die Inhalte der Geodateien.

4.8.3 Persönliche Erfahrungen
Pro Teammitglied ein Kapitel

4.8.3.1 David Tran
Im technischen Bericht (Teil 2).

4.9 Weiterentwicklung
Die nächsten Phasen/Iterationen würde I18N (Internationalization) implementieren. Die OGR2GUI Benutzeroberfläche ist nur Englisch. Als erstes müssten alle englischen Texte in eine Datei geschrieben werden und für jede Sprache eine neue Datei im selber Struktur erstellen.

Ogr2ogr hat eine Standardausgabe, die bei Erfolg oder Fehler eine Meldung aufschreibt. Diese Standardausgabe ist nicht ins GUI implementiert und könnte ein Debugfenster als QDialog werden.

Projektionen sind in OGR2GUI von der Datei gcs.csv von GDAL, aber ogr2ogr kann Projektionen per URL laden. Diese Funktion muss in den Plan vom nächsten Meilenstein.

5 Installationsanleitung

5.1 Teil 1 Computer-Based Training mit Quiz

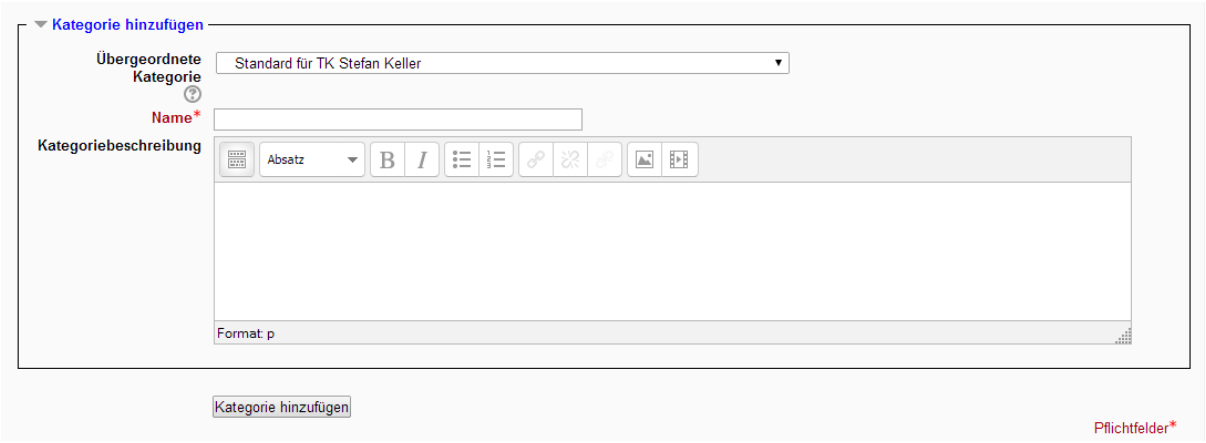
5.1.1 Vorbereitung

Datenbanksysteme Quiz Dateien:

- DB1_1_Datenmodellierung (Domain Modell).txt
- DB1_2_UML-Klassendiagramm, ER-Modell, Martin-Notation, Krähenfussdiagramm.txt
- DB1_3_Relationales Modell (inkl. Abbildung Domain Modell in relationales Modell).txt
- DB1_4_Relationale Algebra und Normalisierung.txt
- DB1_5_SQL Data Definition Language (DDL, PostgreSQL).txt
- DB1_6_SQL Data Manipulation Language (DML, PostgreSQL).txt
- DB1_7_SQL Security (DCL, PostgreSQL).txt
- DB1_8_Transaktionen.txt
- DB1_9_Indexe und Optimierung.txt
- DB1_10_JDBC.txt
- DB1_11_OR Mapping.txt
- DB1_12_PostgreSQL.txt

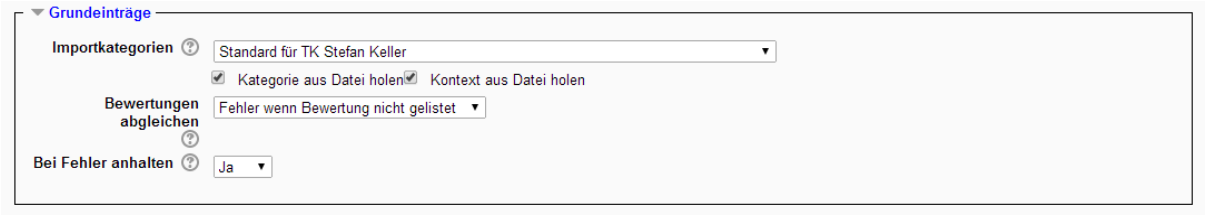
5.1.2 Installation

1. Als Administrator einloggen.
2. Einstellungen -> Fragensammlung -> Kategorie erstellen.



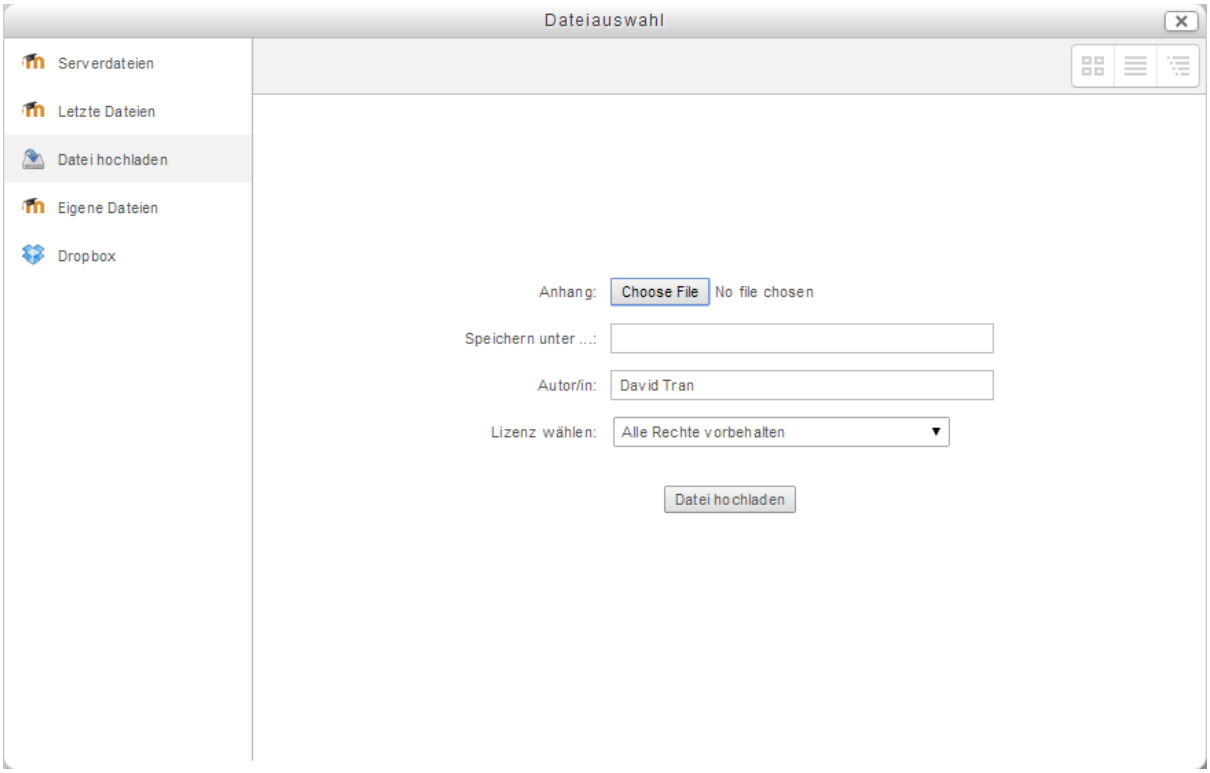
3. Einstellungen -> Fragensammlung -> Import
   1. Dateiformat: Gift-Format

2. Grundeinträge: Importkategorien einstellen



3. Fragen aus Datei importieren: Datei wählen…

3. Datei hochladen: Choose File -> Datei hochladen

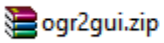


4. Fragen aus Datei importieren: Import

5.2 Teil 2 Reengineering eines Desktop Konverters OGR

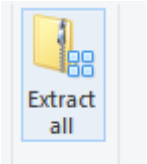
5.2.1 Vorbereitung

OGR2GUI herunterladen.

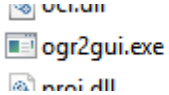


5.2.2 Installation

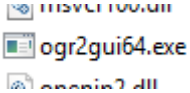
1. OGR2GUI entpacken.



2. ogr2gui.exe ausführen.



3. Empfehlung: ogr2gui64.exe ausführen in Windows x64.



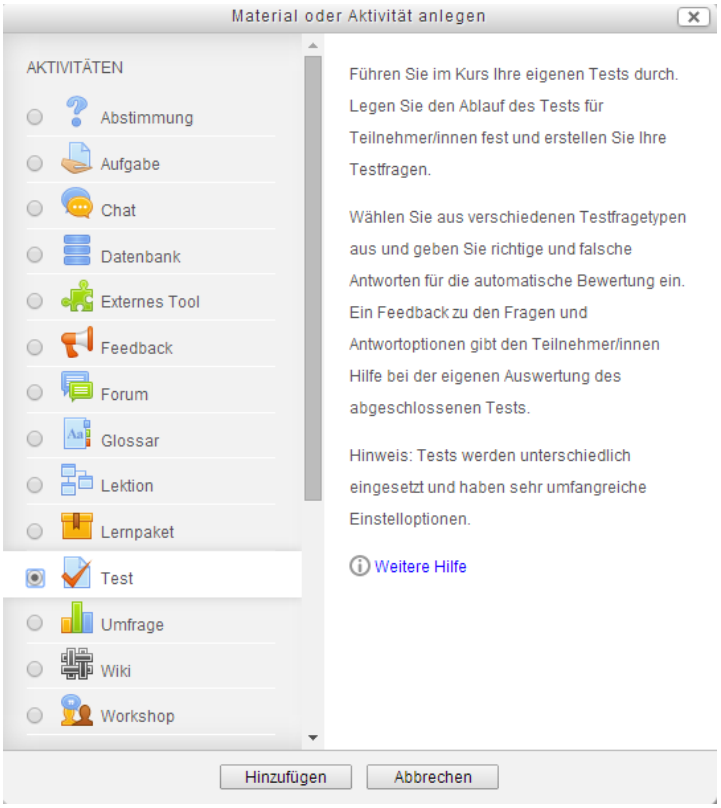
6 Bedienungsanleitung

6.1 Teil 1 Computer-Based Training mit Quiz

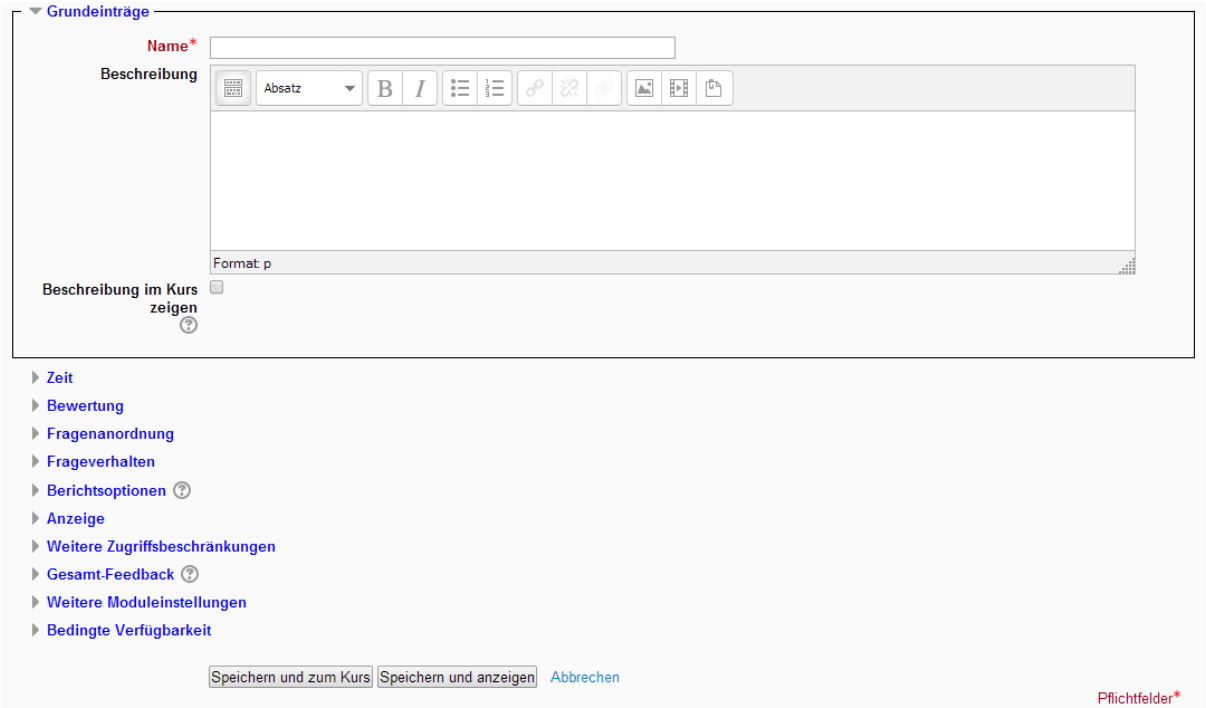
6.1.1 Vorbereitung

1. Unter Navigation einen Kurs auswählen.
2. Bearbeitung einschalten.
3. „Material oder Aktivität anlegen“

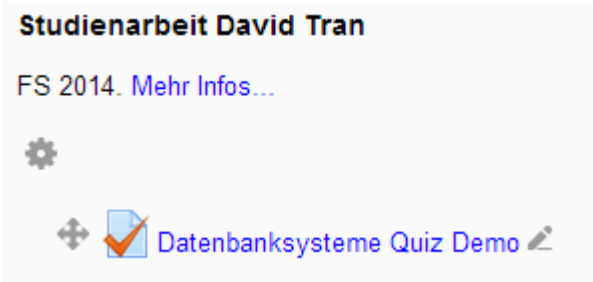
1. Aktivitäten: Test



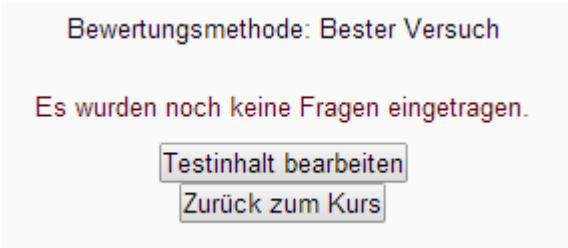
2. Grundeinträge: Name einfügen (z.B. „Datenbanksysteme Quiz Demo“)
3. „Speichern und zum Kurs“



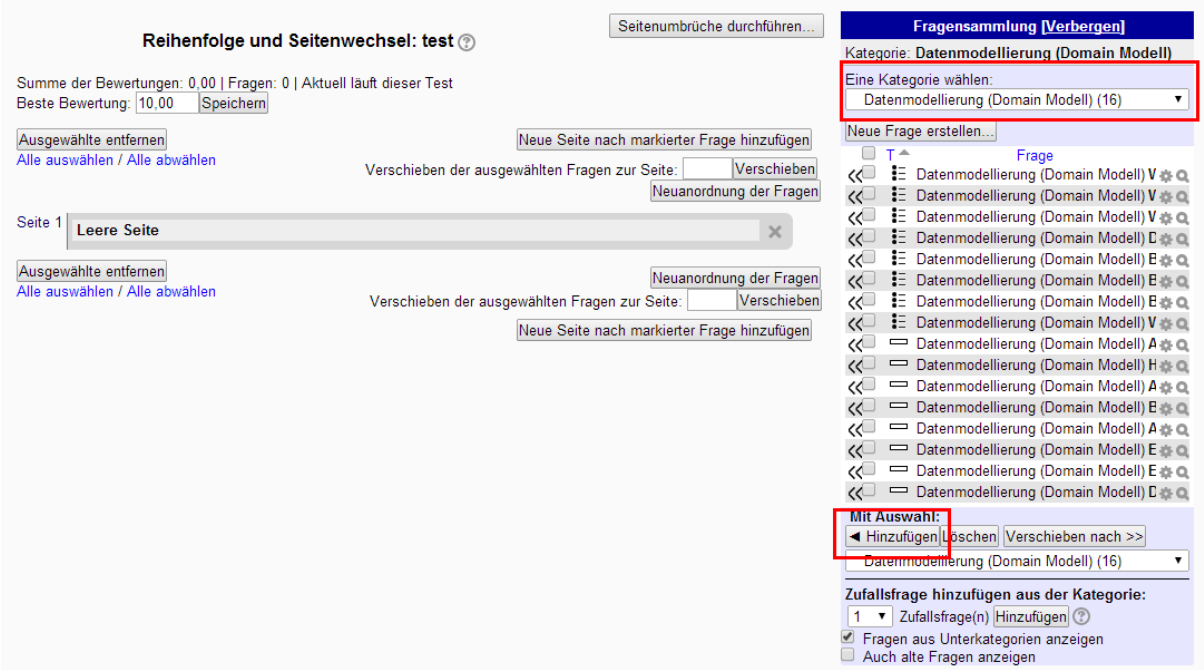
4. „Datenbanksysteme Quiz Demo“



5. Testinhalt bearbeiten

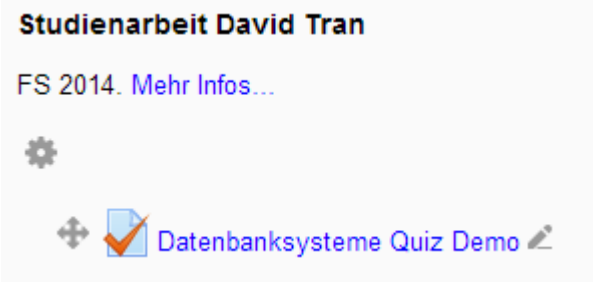


6. Eine Kategorie wählen
7. Mit Auswahl: Hinzufügen

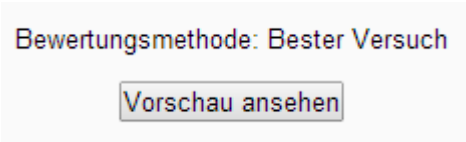


6.1.2 Anwendung

1. Kurs auswählen (z.B. „Datenbanksysteme Quiz Demo“)



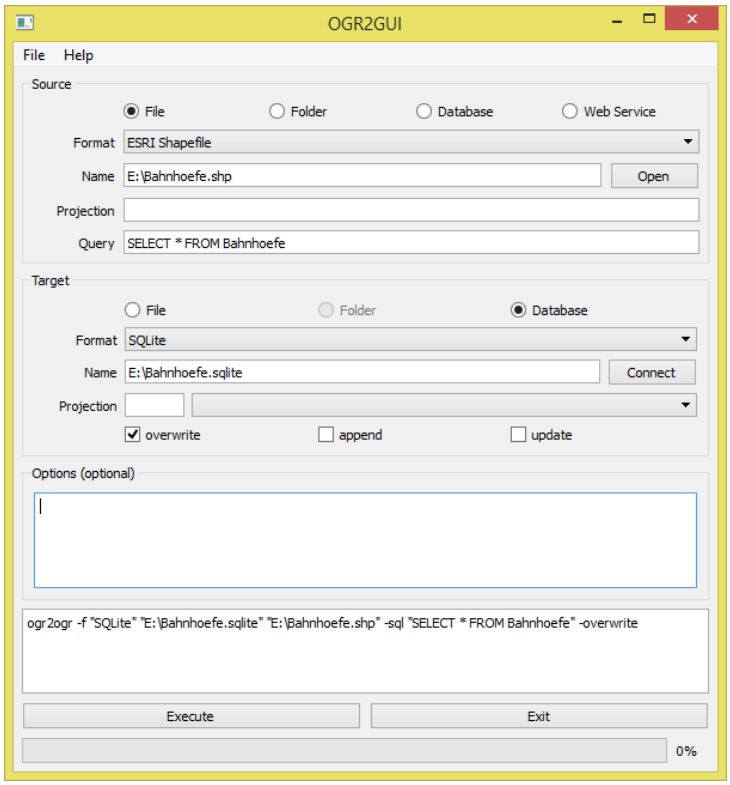
2. Vorschau ansehen



6.2 Teil 2 Reengineering eines Desktop Konverters OGR

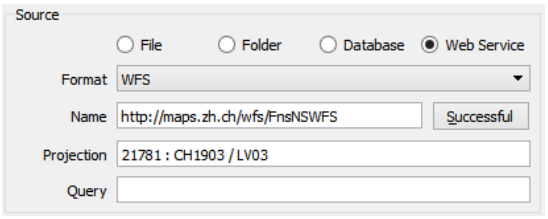
6.2.1 Vorbereitung

OGR2GUI starten:

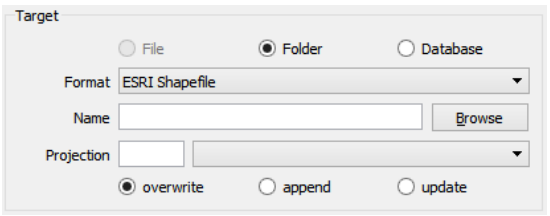


6.2.2 Anwendung

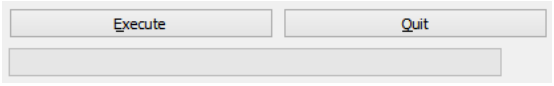
1. Source auswählen.



2. Target auswählen.



3. Konvertierung starten mit „Execute“.



7 Projektplan

7.1 Projekt Übersicht

Es ist im ersten Teil ein Quiz zur Einführung in Datenbanksysteme (vor allem Modul Datenbanksysteme 1) zu erstellen und im zweiten Teil ist die nächste Version des Desktoptool OGR2GUI zur Datenkonversion zu veröffentlichen.

7.1.1 Zweck und Ziel

Das Modul Datenbanksysteme 1 wird neu ab dem Herbstsemester 2014 in das erste Semester vorgelegt und das Quiz auf Moodle (moodle.hsr.ch) soll den Erstsemestrigen beim Selbststudium und der Selbstkontrolle helfen.

Das Tool OGR2GUI (http://www.ogr2gui.ca/) ist mit dem aktuellem Qt Framework (5.2.1) zu aktualisieren, wenn möglich C++11 verwenden und eine vierte Quelle „Web Service“ zu implementieren. Auf https://git.hsr.ch/git/ogr2gui sind die Quelltexte abgelegt.

7.1.2 Lieferumfang

- Quiz (Dbs1)
    - Textdateien (GIFT-Format) für Moodle zum importieren. Inhalt:
        - Multiple Choice
        - Match
        - Lückentext
- OGR2GUI
    - Quelltexte
    - Binäre Datei für die Softwareverteilung

7.1.3 Annahmen und Einschränkungen

- Das Modul Datenbanksysteme 1 ist eine Vorlesung von der Hochschule für Technik Rapperswil. Moodle ist nur für eingeschriebene Studenten, welche sich für das Modul angemeldet haben.
- Die Quiz Textdateien sind im GIFT-Format.
- OGR2GUI ist mit C/C++ und Qt programmiert und ist unter Windows 8.1 aktualisiert. Der Funktionsumfang bleibt bestehen und hat eine neue Quelle „Webservice“.

7.2 Projektorganisation

Es sind zwei unabhängige Aufgaben organisiert: Teil 1 ist nur das Quiz und Teil 2 für OGR2GUI geplant.

7.2.1 Organisationsstruktur

Projektverantwortlicher: David Tran

7.2.2 Externe Schnittstellen

Betreuer: Prof. Stefan Keller

Quelltext Review: Betreuer, Wissenschaftlicher Assistent

7.3 Management Abläufe

7.3.1 Kostenvoranschlag

Insgesamt sind 16 Wochen geplant und mit mindestens 240 Stunden Arbeitsaufwand (pro Woche sind es mindestens 17.14 Stunden).

7.3.2 Zeitliche Planung

Geplant sind vier Phasen mit Iterationen, beginnend ab dem 17. Februar 2014:

- Planung: 0.5 Woche (17.2.2014 – 23.2.2014)
- Teil 1: Quiz ausarbeiten: 5.5 Wochen (24.2.2014 – 28.3.2014)
- Teil 2: OGR2GUI entwickeln: 7 Wochen (31.3.3014 – 6.6.2014)
- Demo und Abgabe: 1 Woche (9.6.2014 – 13.6.2014)

Abgabetermin ist spätestens am ~~30. Mai 2014 um 1700 Uhr~~ (17. März 2014: um zwei Wochen verlängert bis am 13. Juni 2014 um 1700 Uhr).

7.3.2.1 Phasen/Iterationen

Planung/Teil 1 Quiz ausarbeiten (17.2.2014 – 23.2.2014)

W	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
		- „Grob“ Plan erstellen - Zeiterfassung planen - IT-Infrastruktur einrichten															

Teil 1: Quiz ausarbeiten (24.2.2014 – 28.3.2014)

W	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
								- Moodle einarbeiten - Fragen notieren - Quiz erstellen									

Teil 2: OGR2GUI aktualisieren (31.3.2014 – 6.6.2014)

W	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
- Prototypentwicklung - Modultest - Systemtest										Os te rn							

Demo und Abgabe (9.6.2014 – 13.6.2014)

W	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
- Bericht fertigstellen - Demo Quiz und OGR2GUI - Abgabe ist bis am 13.6.14 um 17.00 Uhr																	

7.3.2.2 Meilensteine (MS)

MS1 (17.2.2014 – 21.2.2014)

- Plan erstellen
- Quiz einarbeiten
    - Buch durchblättern
    - Moodle einarbeiten
    - GIFT Format anschauen
- Git Repository einrichten

MS2 (24.2.2014 – 7.3.2014)
- Quiz:
    - Buch lesen
    - Selbstlernfragen aufschreiben (>100 Fragen)
    - Selbstlernfragen sortieren für Multiple Choice, Match und Lückentexte
- OGR2GUI:
    - Qt einarbeiten und die Entwicklungsumgebung kennenlernen.
    - Qt Creator Tutorials anschauen.

MS3 (10.3.2014 – 28.3.2014)
- Quiz:
    - Fragensammlung ins GIFT-Format konvertieren
    - Fragensammlung in Moodle testen.
- OGR2GUI:
    - Quelltexte einarbeiten

MS4 (31.3.2014 – 4.4.2014)
- OGR2GUI:
    - Quelltexte bearbeiten und aktualisieren
    - Webservice einarbeiten

MS5 (7.4.2014 – 2.5.2014)
- OGR2GUI:
    - Prototyp fertigstellen
    - Webservice als neue Quelle hinzufügen und testen

MS6 (5.5.2014 – 23.5.2014)
- OGR2GUI:
    - Fehlerbeseitigung (Tickets)
    - Unit Tests

MS7 (26.5.2014 – 6.6.2014)
- OGR2GUI:
    - Modul- und Systemtests

MS8 (9.6.2014 – 13.6.2014)
- Systemtest und Demonstration von Teil 1 Quiz und Teil 2 OGR2GUI

7.3.3 Besprechungen

Vereinbart wurde mit dem Betreuer, dass wir uns wöchentlich nach Vereinbarung treffen. Die Sitzungen sind protokolliert.

7.4 Risikomanagement

7.4.1 Risiken

ID	Risiko	Auswirkung	Massnahme	Wahrscheinlichkeit [%]	Letzte Aktualisierung
~~Rx~~	~~Probleme beim Umgang mit Moodle~~	~~Einarbeitung planen~~	~~Dokumentation lesen~~	~~5~~	17.3.14

~~Rx~~	~~Noch nie Dateien in GIFT-Format geschrieben~~	~~Grosse und umständliche Textdateien~~	~~Beispiele anschauen~~	~~5~~	17.3.14
~~Rx~~	~~Zuwenig Fragen (<100) für die Selbstkontrolle~~	~~Quiz Fragen wiederholen sich häufig~~	~~Fragen PostgreSQL stellen~~	~~2~~	17.3.14
~~Rx~~	~~Fehlende Ticketsystem~~	~~Arbeitspakete müssen in Excel erfasst werden~~	~~Excel Tabelle mit Arbeitspakete vorbereiten~~	~~1~~	17.3.14
~~Rx~~	~~Quiz Fragen sind zu schwierig~~	~~Ungeeignet für die Selbstkontrolle~~	~~Frageformat ändern~~	~~10~~	26.3.14
~~Rx~~	~~Quiz testen braucht mehr Zeit als geplant~~	~~Einfachere Fragen im Quiz~~	~~Weniger mit QGis Quiz Plugin testen.~~	~~5~~	26.3.14
~~Rx~~	~~Qt Programmierung braucht viel mehr Zeit~~	~~Verzögerung~~	~~Bessere Einführung suchen~~	~~1~~	27.3.14
~~Rx~~	~~Keine Unit Tests mit QT~~	~~Es braucht eine neue Entwicklungsumgebung~~	~~Unit Tests ohne QT implementieren und testen~~	~~1~~	7.4.2014
~~Rx~~	~~Gdal/ogr Bibliothek kompiliert nicht mit allen Datenbanken (OCI, MySQL, ODBC)~~	~~ogr2ogr/ogrinfo Bibliothek geht nur mit SQLite, PostgreSQL~~	~~Gdal Dokumentation lesen und Dev Mailingliste anschreiben~~	~~5~~	10.4.2014
~~Rx~~	~~Vorkompiliertes Gdal nehmen/ Mit binäre Gdal Dateien implementieren~~	~~Verzögerung, weil es neue Quelltexte benötigt.~~	~~Gezielt Vorgehen, Refactoring~~	~~10~~	17.4.2014
~~Rx~~	~~MSVC anstatt MinGW~~	~~Neue API~~	~~Tutorials anschauen~~	~~1~~	17.4.2014
~~Rx~~	~~Probleme bei der Aktualisierung von OGR2GUI~~	~~Verzögerung~~	~~Fragen stellen (Betreuer und co./ Mailingliste)~~	~~10~~	25.4.2014
~~Rx~~	~~OGR C++ API funktioniert nicht~~	~~Quelltexte sind nicht in C++11~~	~~C Programmiersprache verwenden~~	~~5~~	16.5.2014
R1	Quelltext hat viele Smells	Keine Erfahrung mit C Refactoring	Nachschlagewerke benutzen	1	
R2	Quelltexte (Ogr.h, Ogr.cpp) ersetzen (ogr2ogr.cpp)	Qt Quelltexte umschreiben	Quelltexte refaktorisieren	3	
R3	Nicht Dokumentierte Optionenauswahl implementieren (DSCO, LCO usw.)	Höchstens eine Option kann implementiert werden	Ogr2ogr.cpp zum nachschauen benutzen	3	

7.5 Arbeitspakete

Die Arbeitspakete sind im separaten Dokument *Projektplan.xlsx*. Die wöchentliche Zeiterfassung ist im Kapitel *Zeiterfassung* genauer beschrieben.

7.6 Qualitätsmassnahmen

7.6.1 Dokumentation

Die gesamte Dokumentation ist auf Microsoft OneDrive und owncloud.hsr.ch synchronisiert. Auf owcloud.hsr.ch hat der Betreuer Leserechte.

7.6.2 Projektmanagement

Arbeitspakete sind weiter oben aufgelistet.

7.6.3 Entwicklung

Quelltexte von OGR2GUI sind auf dem internen Git-Server abgelegt (git.hsr.ch). Der Betreuer und wissenschaftliche Angestellte haben Leserechte.

7.6.3.1 Vorgehen

1. Die Quelltexte sind mit Qt 5.2.1 aktualisiert.
2. OGR2GUI Quelltexte sind mit dem aktuellsten C-Compiler kompiliert.
3. Die neue Quelle Webservice ist implementiert.

7.6.3.2 Code Reviews

Der Quelltextreview von OGR2GUI ist mit dem Betreuer oder einem wissenschaftlichen Assistenten.

Projektplan.xlsx

Arbeitspakete:

Arbeitspakete	Zeit [h]	SOLL	IST	Meilensteine	Phase/ Iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Buch lesen	8	8	8	MS2	*		8															
Meetings	18	14	21,75	*	*	1	1	1,5	1	1	1,75	1,5	1,5	0,5	2	1,5	1	1,5		1,5	2	1,5
Zwischentotal	26	22	29,75																			
Projektplan	11	11	10	MS1	Planung	10																
Git	1	1	0,1	MS1	- Planung	0,1																
					- OGR2GUI aktualisieren																	
Dokumentationsvorlagen	2	1	2	MS1	Planung	2																
GIFT-Textformat einarbeiten	2	2	2	MS1	Planung	2																
Risikomanagement	2	2	2	MS1	Planung	2																
Zwischentotal	18	17	16,1																			
Moodle Login besorgen	0,5	0,5	0,1	MS2	Quiz ausarbeiten	0,1																
Moodle einarbeiten	4	4	3,5	MS2	Quiz ausarbeiten	1		2	0,5													
Fragen vorbereiten	25	20	21	MS2	Quiz ausarbeiten		7	10	3	1												
Fragen für Frageformate sortieren	4	2	5	MS2	Quiz ausarbeiten			5														
Fragen in GIFT-Textformat umwandeln	4	10	12	MS3	Quiz ausarbeiten			4	8													
GIFT in Moodle importieren	0,5	0,5	0,5	MS3	Quiz ausarbeiten				0,5													
GIFT in Moodle testen	1	4	6	MS3	Quiz ausarbeiten					4	2											
GIFT in QGis Quiz Plugin testen	1	2	4,5	MS3	Quiz ausarbeiten						2			0,5		2						
ogr2ogr Kommandozeile und WFS anschauen	1	0,5	1	MS3	- OGR2GUI aktualisieren				0,5		0,5											
Git einarbeiten/ Beispiele anschauen/ Tutorial	2	1	2	MS2	- Quiz ausarbeiten		0,5		1	0,5												
					- OGR2GUI aktualisieren																	
OGR2GUI einarbeiten/ Webservice einarbeiten	17	15	17	MS3/MS4	- Quiz ausarbeiten				4	9	4											
					- OGR2GUI aktualisieren																	
Dokumentation Einleitung/ Vision	8	9	11	*	OGR2GUI aktualisieren		8		2									1				
Dokumentation Analyse	4	4	5,5	*	OGR2GUI aktualisieren											5,5						
Zwischentotal	72	72,5	89,1																			
Quelltext Review	5	3	3	MS4-MS8	OGR2GUI aktualisieren																3	
Quelltext Refactoring	10	10	15,5	MS5	OGR2GUI aktualisieren								3	2		2,5	2	0,5	2	0,5	3	
Qt Framework installieren	1	1	1	MS2	OGR2GUI aktualisieren						1											
OGR2GUI aktualisieren	27	33	43,5	MS3-MS6	OGR2GUI aktualisieren						9	5	2		9	2		3,5	1	12		
Mingw-builds Qt und Compiler einrichten	5	5	5	MS5	OGR2GUI aktualisieren								5									
Visual Studio einarbeiten	4	4	9	MS5	OGR2GUI aktualisieren								4,5	4,5								
Webservice hinzufügen	15	15	20	MS5/MS6	OGR2GUI aktualisieren								4	9	2					5		
Unit Tests	2	2	3,5	MS6	OGR2GUI aktualisieren											0,5	3					
Optionen/ Flags hinzufügen (LCO, DSCO)	8	8	26,5	MS6	OGR2GUI aktualisieren													11,5	15			
Modultests	4	4	4,5	MS7	OGR2GUI aktualisieren										0,5		4					
Systemtests	4	4	4,5	MS7	OGR2GUI aktualisieren										0,5							
Dokumentation Anforderungen	8	8	17,5	*	OGR2GUI aktualisieren							6			4	4				4		0,5
Dokumentation Ergebnisse	8	8	7,5	*	OGR2GUI aktualisieren					3		2				1			1			0,5
Dokumentation Design	8	8	9	*	OGR2GUI aktualisieren							0,5					8					0,5
Dokumentation Implementation	8	8	9,5	*	OGR2GUI aktualisieren							1								2	6	0,5
Dokumentation Test	4	4	5,5	*	OGR2GUI aktualisieren							2									3	0,5
Dokumentation Anhang (Abstract/ Management Summary)	4	4	13,5	MS7	OGR2GUI aktualisieren																10	3,5
Zwischentotal	125	129	198,5																			
Plan nachführen	4	4	3,4	MS8	Demo und Abgabe	0,5	0,5	0,2	0,1	0,2	0,1		0,1	0,1	0,1	0,1	0,5	0,1	0,1	0,1	0,1	0,5
Systemtest	4	4	4	MS8	Demo und Abgabe																	4
Demo Quiz und OGR2GUI	1	0,5	0,5	MS8	Demo und Abgabe																	0,5
Softwareverteilung	1	1	1	MS8	Demo und Abgabe																	1
Dokumentation drucken und binden	2	1	1	MS8	Demo und Abgabe																	1
Dokumentation zur Kontrolle abgeben (Kurzfassung/Poster)	0,5	0,5	2,5	MS8	Demo und Abgabe																	2,5
Zwischentotal	12,5	11	12,4			18,7	25	22,7	20,6	18,7	20,35	18	20,1	19,6	18,1	19,1	18,5	18,1	19,1	25,1	27,1	17
Insgesamt	253,5	251,5	345,85		**Subtotal/ KW**	276,65																

Tickets:

Tickets

Datum	Erledigt	Beschreibung	Kommentar	Nummer	Legende
07.04.2014	07.04.2014	File als Source zeigt beim Öffnen-Dialog alle Dateien an	siehe 0.6	1	TODO
07.04.2014	07.04.2014	File als Target zeigt beim Speichern-Dialog alle Dateien an	siehe 0.6	2	OPEN
07.04.2014	25.04.2014	~~MySQL, OCI, ODBC Datenbanken fehlen in der eigenen Gdal Bibliothek~~	MySQL kompiliert mit Mingw-builds nicht	3	FINISH
07.04.2014	14.04.2014	Web Service Connect Button testet die Onlineverbindung	Verbindungstest beim Einfügen einer URL	4	MESSAGE
07.04.2014	07.04.2014	QTextEdit ist editierbar		5	
07.04.2014	07.04.2014	OGR Formate - Dateiendungen herausfinden	Dateiendungen Dokument erhalten	6	
07.04.2014	05.05.2014	QTextEdit Ausgabe übergibt OGR2OGR alle Flags	siehe 16/17	7	
07.04.2014	17.04.2014	Gdal Bibliothek mit neue OGR Formate kompilieren und testen	Mingw-builds kompatible Plugins	8	
07.04.2014	14.04.2014	Web Service integrieren	Nur default Layer	9	
07.04.2014	17.04.2014	OGR2GUI mit Visual Studio 2012 kompilieren		10	
07.04.2014	17.04.2014	OGR Formate von einer Datei auslesen anstatt von Dta.h	OGR2GUI oder X2GUI???	11	
08.04.2014	22.05.2014	Logfenster verbessern		12	
09.04.2014		~~OpenSSL 1.0.1g mit PostgreSQL neu kompilieren~~	OpenSSL: TLS Heartbeat Bugfix	13	
09.04.2014	24.04.2014	SQLite Datenbank als "File" Refaktorisieren	SQLite braucht kein Datenbankverbindungsaufbau	14	
09.04.2014	09.04.2014	flag --without-libtool error von gdal 1.9.2-1.10.1 als Ticket auf trac.osgeo.org		15	
10.04.2014	22.05.2014	flag -overwrite, -append, -update implementieren, weil es nicht implementiert ist		16	
10.04.2014	15.05.2014	Layer Creation Option (LCO), Dataset Layer Option (DCSO)		17	
14.04.2014	14.04.2014	Bei Web Service zeigt der Button den Status der WFS Verbindung an		18	
14.04.2014	17.04.2014	Gdal/Ogr und alle integrierbare Treiber mit Visual Studio kompilieren	Vorkompilierte Gdal Bibliothek (MSVC 2010)	19	
15.04.2014	16.04.2014	WFS soll alle Layers herunterladen		20	
15.04.2014	24.04.2014	Progressbar läuft nach der Konvertierung weiter		21	
16.04.2014	02.06.2014	Konvertierung mit Threads erweitern		22	
16.04.2014	05.05.2014	SQLiteverbindung und -konvertierung testen/implementieren		23	
17.04.2014	24.04.2014	WFS -> GMT ist der Speicherort falsch	Ticket auf trac.osgeo.org	24	
17.04.2014	17.04.2014	OGR2GUI mit Visual Studio 2010 kompilieren (x86 & x64)	Nach fehlenden DLLs suchen	25	
17.04.2014	25.04.2014	SE Modell dokumentieren		26	
22.04.2014	07.05.2014	Unit Tests für OGR2GUI schreiben		27	
22.04.2014	22.04.2014	Fehlende DLLs für die binäre Datei suchen		28	
22.04.2014	22.04.2014	Separate *.pro Dateien	Toolchain & Architektur separiert	29	
22.04.2014	22.04.2014	Formate in die Datei Dta.h mit ogrinfo --formats synchronisiert		30	
22.04.2014	22.04.2014	QSqlDatabase: QPSQL driver not loaded (MSVC x64)		31	
22.04.2014	24.04.2014	Target: Projektionssuchfeld soll nur Zahlen erlauben		32	
24.04.2014	24.04.2014	ogrinfo PostgreSQL Verbindungsaufbau	MSVC & Mingw builds	33	
25.04.2014	13.05.2014	ogr2ogr.cpp Quelltext einarbeiten		34	
01.05.2014	07.05.2014	Qdoc mit Doxygen (Konfigurationsdatei)		35	
05.05.2014	22.05.2014	SQLite Datenbank soll mehrere Layers speichern		36	
07.05.2014	28.05.2014	Systemtest ohne ogr2ogr.cpp		37	
07.05.2014	07.05.2014	Fix: slash für Windows OS		38	
07.05.2014	07.05.2014	Disable Optionen: overwrite, append, update		39	
12.05.2014	13.05.2014	WFS Fehler in GDAL 1.11: https://github.com/gisinternals/buildsystem/issues/65	1.10.0 DLLs benutzen: libcurl.dll, libeay32.dll, ssleay32.dll	40	
12.05.2014	13.05.2014	GDAL 1.11 updaten		41	
12.05.2014	13.05.2014	Optionenfenster oberhalb des Logfensters		42	
13.05.2014	13.05.2014	DSCO implementiert		43	
15.05.2014	26.05.2014	Optionen (DSCO, LCO usw.) im Logfenster darstellen		44	
15.05.2014	22.05.2014	Layer Creation Option (LCO) Option implementieren		45	
19.05.2014	22.05.2014	ogr2ogr.cpp ins OGR2GUI hinzufügen		46	
21.05.2014	22.05.2014	OGR2GUI Quelltext Doc im selben Ordner verlinken wie ogr2gui.exe		47	
21.05.2014	27.05.2014	Projektionen aus gcs.csv lesen		48	
19.05.2014	21.05.2014	Readme aktualisieren mit einer Installationsanleitung		49	
21.05.2014	21.05.2014	Zu ogr2gui das ogr2ogr Kommandozeile Tool einfügen		50	
22.05.2014	09.06.2014	Doc generieren		51	
22.05.2014	26.05.2014	WFS downloads mit Layerauswahl		52	
26.05.2014	09.06.2014	~~GUI: DSCO, LCO, Other options Implementieren~~		53	
26.05.2014	26.05.2014	Menu Shortcuts		54	
26.05.2014	26.05.2014	Help neu verlinken		55	
26.05.2014	02.06.2014	Command Line Fenster bei release deaktivieren	per Menüeintrag ein-/ausschalten	56	
26.05.2014	26.05.2014	WFS Bugfix		57	
26.05.2014	27.05.2014	~~Qt 5.3 update~~	PostgreSQL (QSQL) geht nicht	58	
26.05.2014	26.05.2014	Überflüssige *.csv im Ordner data löschen		59	
26.05.2014	28.05.2014	Im Optionenfenster werden newlines nicht erkannt		60	
27.05.2014	28.05.2014	SQLite und QSLite genauer anschauen		61	
27.05.2014	28.05.2014	Logfenster, Progressbar: Fortschritte anzeigen		62	
27.05.2014	28.05.2014	Testdateien für Tests erstellen		63	
29.05.2014	29.05.2014	Fehlende SQL DLLs für den Standalone betrieb		64	
29.05.2014	29.05.2014	*.pro x86 und x64 zusammengefasst in eine ogr2gui.pro Datei		65	
29.05.2014	29.05.2014	Options input fehlt		66	
29.05.2014	29.05.2014	OGR API beim öffnen von einer Source		67	
30.05.2014	02.06.2014	CreateProcess mit Qprocess ersetzen		68	
30.05.2014		Qprocess output in QTextEdit umleiten		69	
30.05.2014		Debug Fenster erstellen		70	
30.05.2014		Menüeintrag um das Debug Fenster ein-/auszuschalten		71	
04.06.2014		~~Threadpool implementieren~~		72	
09.06.2014	09.06.2014	Qprocess stdoutput in eine Datei speichern		73	
09.06.2014	10.06.2014	Genauere Qt-Tests		74	
10.06.2014	11.06.2014	Kompatibilität mit Win32, Win64 und Wow64		75	
11.06.2014	11.06.2014	binäre Dateien kompilieren		76	

8 Zeiterfassung

Beschreibt den gesamten Aufwand in Stunden, welcher für jede Iteration jeweils gebraucht wurde. Das Protokoll listet die Tätigkeiten detailliert auf.

8.1 Planung/Quiz ausarbeiten (17.2.2014 – 23.2.2014)

	18h	18h/17h

Total: 18 Stunden

Protokoll

Datum	*Arbeitszeit (h)*	*Bemerkung*
17.2.2014	9	- Dokumentationsvorlagen vorbereitet. - Plan erstellt. - GIFT-Format angeschaut. (http://docs.moodle.org/26/en/GIFT_format)
18.2.2014	8	- Plan überarbeitet und ausgedruckt. - Buch durchgeblättert. - Email an Betreuer gemäss Projektorganisation. - Meeting mit dem Betreuer.
19.2.2014	1	- Meeting Protokoll ins Word Dokument eingetragen. - Korrekturen im Projektplan bei der Zeitberechnung. (Plan nachgeführt)

8.2 Teil 1: Quiz ausarbeiten (24.2.2014 – 28.3.2014)

	25h	22.5h	20.5h	19h	20h	107h/85h

Sub Total: 125 Stunden

Protokoll

Datum	*Arbeitszeit (h)*	*Bemerkung*
24.2.2014	9	- Buch lesen. - 50% mögliche Selbstlernlösungen für das Quiz aufgeschrieben. - Email an Betreuer gemäss Projektorganisation. - Qt 5.2.1 installiert.
25.2.2014	8	- Meeting mit dem Betreuer. - Meeting Protokoll ins Word Dokument eingetragen. - 50% mögliche Selbstlernlösungen für das Quiz aufgeschrieben.
26.2.2014	8	- Plan nachgeführt. - Qt Einführung. - Bericht Einleitung geschrieben.
3.3.2014	8	- Selbstlernlösungen sortieren. - Fragen vorbereiten. - Moodle Fragensammlung eingerichtet. - Beispielsfragesammlung mit vier Fragetypen importiert.
4.3.2014	8	- Email an Betreuer gemäss Projektorganisation. - Bücher auf Google Books als Bookmark.

5.3.2014	6.5	- Multiple Choice Fragensammlung konvertiert. - Meeting mit dem Betreuer. - Multiple Choice Fragen fertigstellen.
10.3.2014 11.3.2014 12.3.2014	9 8.5 3	- Multiple Choice Fragen Demo in Moodle eingerichtet. - Match und Lückentexte Fragensammlung konvertieren. - Matching Fragen Demo in Moodle eingerichtet. - Lückentextfragen Demo in Moodle eingerichtet. - OGR2GUI mit Qt geöffnet: Datei ogr_api.h fehlt. - QGis Quiz ist der Fragetyp Lückentext unvollständig. - PostgreSQL Selbstlernfragen erstellen. - PostgreSQL Selbstlernfragen in GIFT konvertieren. - Qt ogr2gui.pro OSGeo4W64 Quelltexte hinzugefügt. - Qt OGR2GUI kompiliert nicht (MinGW???). - Email an Betreuer gemäss Projektorganisation. - Quiz nach Kategorien neu sortiert. - Bericht Ergebnisse von Teil 1 geschrieben. - Meeting mit dem Betreuer.
17.3.2014 18.3.2014 19.3.2014	8 7 4	- Moodle Fragensammlung testen. - Bericht Ergebnisse von Teil 1 aktualisiert. - OGR2GUI mit Qt 5.2.1 weiter recherchieren (Qt4 to Qt5). - Projektplan aktualisiert (Abgabetermin: 17.6.14). - Bericht Einleitung von Teil 2 aktualisiert. - Email an Betreuer gemäss Projektorganisation. - gdal, proj, postgresql mit Cygwin kompilieren. - Meeting mit dem Betreuer.
24.3.2014 26.3.2014 27.3.2014	9 9.5 1.75	- Qt Programmierumgebung ist aufgesetzt. - OGR2GUI benutzte Datenbanken (MySQL, PostgreSQL, SQLite einarbeiten. - OGR2GUI Grafische Benutzeroberfläche anzeigen. - OGR2GUI Radio Button (WFS) hinzufügen. - gdal, proj, postgresql mit MinGW kompilieren. - Email an Betreuer und co. bzgl. Gdal kompilieren. - Quiz mit Moodle und QGis Quiz Plugin testen. - Bibliotheken neu kompilieren und testen. - Gdal/ Ogr Bibliothek anschauen. - Email an Betreuer gemäss Projektorganisation. - Meeting mit dem Betreuer.

8.3 Teil 2: OGR2GUI aktualisieren (31.3.2014 – 6.6.2014)

202h/170h	18h	20h	19.5h	18h	19h	18h	18h	19h	25.5h	27h	

Sub Total: 327 Stunden

Protokoll

Datum	*Arbeitszeit (h)*	*Bemerkung*
31.3.2014 2.4.2014	9.5 7	- SE Dokumentation Teil 1 aktualisiert. - Gdal SVN stable 28.4.2014 kompiliert (ogrinfo/ogr2ogr 1.9.2). Es fehlen diese Datenbanken: MySQL, OCI, ODBC. Gdal 1.10.x kompiliert nicht mit denselben Flags. - ogrinfo --formats, ogr2ogr --version sind in Ordnung. - SE Dokumentation Teil 1 aktualisiert.

		- Email an Betreuer und co. bzgl unixODBC und MySQL kompilieren mit MinGW. - Email an Betreuer gemäss Projektorganisation.
3.4.2014	1.5	- Meeting mit dem Betreuer.
7.4.2014	10	- Aktuellste Bibliotheken mit Mingw-builds kompiliert. - Mingw-build QtSDK-i686 als neue Entwicklungsumgebung aufgesetzt. - Microsoft Visual Studio 2010 installiert. - Qt MSVC installiert. - Bugfix: Öffnen-Dialog filtert per Dateiendung. - Bugfix: Speichern-Dialog filtert per Dateiendung. - OGR2GUI Textausgabe Widget ist editierbar. - Ticketing in der Excel Tabelle „Projektplan.“
9.4.2014	8	- OGR2GUI OCI Target gelöscht. - OGR Implementierung angeschaut. - Web Service Implementation überlegt. - MySQL Kompilierungen versucht. - Email an Betreuer gemäss Projektorganisation.
10.4.2014	2	- Übersichtstabelle OGR2GUI, X2GUI, MSVC. - Visual Studio 2012 installiert und Qt (MSVC). - Meeting mit dem Betreuer.
14.4.2014	9	- Gdal MSVC kompilieren. - WFS Onlineverbindung.
16.4.2014	8.5	- WFS implementiert. - Letzte Veröffentlichung von QGis Quiz Plugin testen. - SE Dokumentation Teil 2 aktualisieren. - Email an Betreuer gemäss Projektorganisation.
17.4.2014	2	- Visual Studio 2010 installiert und Qt (MSVC x86/x64) - Meeting mit dem Betreuer.
22.4.2014	8	- Vorkompilierte Gdal Bibliothek von gisinternals verwenden. - OGR2GUI mit Visual Studio 2010 kompilieren. - Separierte *.pro Dateien (Mingw-builds, MSVC 2010 x86 und x64). - ogrinfo --formats Log in eine Datei speichern. - Formate in die Datei Dta.h mit ogrinfo --formats synchronisiert - Fehlende DLLs für die binäre Datei hinzufügen. - SE Dokumentation Teil 2 aktualisieren. - Plan nachgeführt.
24.4.2014	7	- PostgreSQL Datenbankverbindungtest (ogrinfo, OGR2GUI). - SQLite Datenbank als Datei öffnen/speichern. - Target Projektionssuchfeld sind nur Zahlen von 0 bis 99999 gültig. - SE Dokumentation Teil 2 aktualisieren. - Email an Betreuer gemäss Projektorganisation.
25.4.2014	3	- ogr2ogr.cpp einarbeiten. - Meeting mit dem Betreuer.
28.4.2014	8	- SE Dokumentation Teil 2 aktualisieren. - ogr2ogr.cpp refaktorisieren.
30.4.2014	8.5	- SE Dokumentation Teil 2 aktualisieren. - ogr2ogr.cpp refaktorisieren. - Letzte Veröffentlichung von QGis Quiz Plugin testen. - Email an Betreuer gemäss Projektorganisation.

1.5.2014	2,5	- Bericht Ergebnisse von Teil 2 schreiben. - Meeting mit dem Betreuer.
5.5.2014	8	- SE Dokumentation Teil 2 aktualisieren.
6.5.2014	7	- Quelltexte refaktorisiert (main.cpp, testmain.cpp) - Unit Test implementieren. - Modul Test implementieren und durchführen.
7.5.2014	2	- Slash für Windows OS korrigiert. - Optionen: overwrite, append, update deaktiviert - Gemäss Email: o ogr2ogr.cpp Code in- und auswendig. o ogr2ogr-Parameter. - Email an Betreuer gemäss Projektorganisation.
8.5.2014	1	- Meeting mit dem Betreuer.
12.5.2014	8	- Projektplan aktualisiert (+2 Wochen). - GDAL 1.11 funktioniert WFS nicht. - Vorteile/Nachteile für den Benutzer bzgl. Ogr2gui/ Wrapper/ Qt-GUI ist im technischen Bericht. - DSCO, LCO implementieren. - WFS Fehler in GDAL 1.11 gemeldet https://github.com/gisinternals/buildsystem/issues/65
13.5.2014	7.5	- Optionenfenster (Tabelle) oberhalb des Logfensters implementieren. - DSCO implementiert. - GDAL 1.11 mit CURL, SSDL DLLs von 1.10.0. - Formateliste mit GDAL 1.11 synchronisiert. - OGR API C++ geht nicht. - Dokumente bis 27.5.2014???
15.5.2014		- Email an Betreuer gemäss Projektorganisation.
16.5.2014	2.5	- Ogr2ogr.cpp hinzugefügt. - Meeting mit dem Betreuer.
19.5.2014	7.5	- Ogr2ogr.cpp Quelltext ins OGR2GUI eingefügen.
20.5.2014	7.5	- Ogr2ogr.cpp Quelltext ins OGR2GUI eingefügen.
21.5.2014	4	- Technischer Bericht Teil 2 fertigstellen. - Ogr2ogr.cpp als Kommandzeile in OGR2GUI hinzufügen. - Logfenster aktualisiert bei jeder Änderung im GUI. - Optionen aktiviert (z.B. overwrite, append, update, dsco, lco usw.) - Doc verlinkt mit einer lokalen index.html. - Readme hat eine Installationsanleitung.
26.5.2014	7.5	- Optionen sind auch im Logfenster dargestellt. - WFS zeigt eine Layerauswahl an (Klasse WFSConnect). - Menü haben Shortcuts. - Menütexte sind umbenannt. - „Help“ Menü sind neu verlinkt. - Überflüssige *.csv im Ordner data sind gelöscht. - Kein Qt 5.3 Update, weil PostgreSQL (QPSQL) Treiber im Qt Creator nicht erkannt werden. - Doxyfile erstellt. - ogr2ogrThread Klasse erstellt. - Email an Betreuer gemäss Projektorganisation.
27.5.2014	6	- Projektionen aus gcs.csv lesen. - Meeting mit dem Betreuer.
28.5.2014	6	- Optionenfenster kürzt Leerschläge zu einem Leerschlag. - SQLite zeigt in einem Dialogfenster die Tabellen an, ohne

		sie aber auswählen zu können, weil GDAL keine Tabellenauswahl unterstützt. - Logfenster und Progressbar zeigt den Fortschritt an.
29.5.2014	6	- SE Dokumentation Teil 2 weiterführen. - Systemtests mit Ogr API. - Fehlende SQL DLLs hinzugefügt. - *.pro x86 und x64 zusammengefasst in eine ogr2gui.pro Datei. - Bugfix: Optionen Input - Check mit OGR API beim Öffnen einer Source. - Testdateien für Tests erstellt (shp, sqlite).
2.6.2014	5	- SE Dokumentation Teil 2 weiterführen.
3.6.2014	15	- SE Dokumentation Teil 2 fertigstellen. - Quelltext Refactoring/Review.
4.6.2014	5	- Abstract schreiben.
5.6.2014	5	- Management Summary schreiben. - Poster erstellt. - Email an Betreuer gemäss Projektorganisation.
6.6.2014	2	- Meeting mit dem Betreuer.

8.4 Demo (9.6.2014 – 13.6.2014)

17h/17h	17h

Gesamttotal: 344 Stunden

Protokoll

Datum	*Arbeitszeit (h)*	*Bemerkung*
9.6.2014	5	- Abgabe Kurzfassung und des A0-Posters per E-Mail an den Betreuer zur Kontrolle. - Bericht für den Druck vorbereiten. - Email an Betreuer gemäss Projektorganisation.
10.6.2014	5	- Bericht korrigieren. - Qt-Tests ergänzen.
11.6.2014	7	- Meeting mit dem Betreuer. - Abgabe vorbereiten. - Softwareverteilung vorbereiten. - CD vorbereiten. - CD-Inhalt dokumentieren. - Abgabe der vom Betreuer freigegebenen Kurzfassung **als Word-Dokument** per Email an das Studiengangsekretariat. - Abgabe CD für Archivierung und Publikation auf eprints.hsr.ch - Bericht drucken und binden.

Abgabe bis 13.6.2014 um 17:00 Uhr.

Kapitel III Dokumente

9 Infrastruktur

9.1 owncloud.hsr.ch

Es ist eine persönliche Dateiablage, die man mit anderen HSR Angehörige teilen kann. Die Clientsoftware synchronisiert regelmässig mit dem Server, dabei sind alle Dateien auf dem Client und auf dem Server gespeichert.

9.2 Git

Git ist eine Open Source Versionsverwaltung für Quelltexte. Es muss immer ein Master Branch existieren.

10 Glossar

Dbs1	Datenbanksysteme 1
DDL	Data Definition Language
DML	Data Manipulation Language
DCL	Data Control Language
GCC	GNU Compiler Collection
GDAL	Geospatial Data Abstraction Library
GIS	Geoinformationssystem
JDBC	Java Database Connectivity
MSVC	Microsoft Visual C/C++
RUP	Rational Unified Process
SQL	Structured Query Language
SRS	Spatial Reference System
UML	Unified Modeling Language
URI	Uniform Resource Identifier
WFS	Web Feature Service

11 Literaturverzeichnis

[1] Faeskorn-Woyke, H (2007). Datenbanksysteme - Theorie und Praxis mit SQL2003, Oracle und MySQL (1. Auflage). München: Pearson Studium.

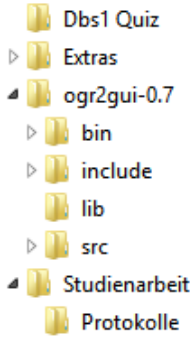
[2] http://docs.moodle.org/26/de/GIFT-Format (26.2.2014)

[3] http://de.wikipedia.org/wiki/Moodle (25.4.14)

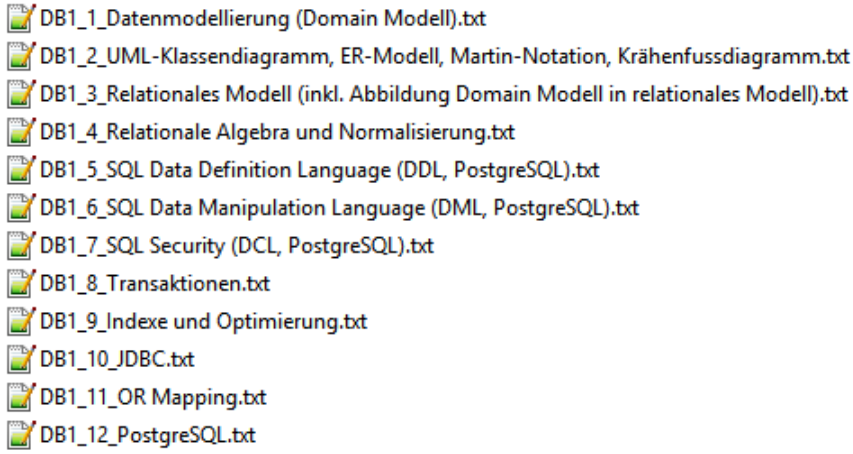
[4] http://www.geo.admin.ch/internet/geoportal/de/home/topics/geobasedata/FAQ/Geobasisdaten_des_Bundesrechts.html (11.6.2014)

12 CD-Inhalt

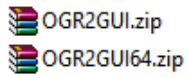
- Ordnerstruktur



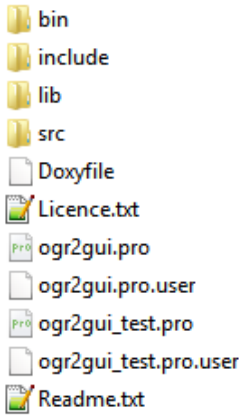
- Dbs1 Quiz:



- Extras:



- ogr2gui-0.7:



- Studienarbeit: