# GWT BASIERTES SYSTEM ZUR AUTOMATISCHEN BEWERTUNG VON PROGRAMMIERAUFGABEN

Daniel Eggert und Eike Clas Schulze

Institut für Kartographie und Geoinformatik, Leibniz Universität Hannover
eggert@ikg.uni-hannover.de

**ABSTRACT**
Neben der Vermittlung wichtiger mathematischer Grundlagen spielt auch die Programmierausbildung im Bachelorstudiengang Geodäsie und Geoinformatik eine Schlüsselrolle. Im Rahmen der Veranstaltung „GIS I / Geländemodellierung“ erfolgt die Programmierausbildung mittels der Programmiersprache Java. Der Fokus liegt, nach einer allgemeinen Einführung, auf der Modellierung und Umsetzung von Datenstrukturen für Geodaten sowie darauf basierenden Algorithmen. Aufgrund der stetig wachsenden Bedeutung von Programmierfähigkeiten wurde beschlossen die Programmierausbildung zu intensivieren, ohne jedoch den Lehr- und Betreuungsaufwand signifikant zu steigern. Eine Lösung stellt ein System zur automatischen Bewertung von Programmieraufgaben dar, da ein solches System einen nahezu konstanten Aufwand unabhängig von der Anzahl der Studierenden aufweist. Dieses Paper gibt einen Überblick über das eigens für diesen Zweck entwickelte System, welches zurzeit in der Programmierausbildung des Institutes eingesetzt wird.

## 1. Einleitung

In vielen nicht ausschließlich technischen Fachbereichen ist das Erlernen mindestens einer Programmiersprache elementarer Bestandteil der Lehre. Der wichtigste Bestandteil einer jeden Programmierausbildung ist das selbstständige Bearbeiten von Programmieraufgaben durch die Studenten. Nur dadurch lässt sich ein nachhaltiger Lerneffekt erzielen. Je nach Größe des Studiengangs, bzw. Art und Umfang der Programmieraufgaben führt dies zu einem erheblichen Betreuungsaufwand für die Lehrenden.

Systeme zur automatischen Bewertung von Programmieraufgaben können den Lehrenden bei dieser Aufgabe unterstützen und den Korrekturaufwand verringern. Darüber hinaus können solche Systeme jedem Studenten ein detailliertes Feedback geben, was die Qualität der Programmierausbildung erhöht. Je nach Zielgruppe und Art der Programmierausbildung sind unterschiedliche Systeme denkbar. Die Spannweite reicht von einfachen Tools die lediglich Teilaufgaben der Bewertung automatisieren bis hin zu vollautomatischen Bewertungssystemen. [1] gibt dahingehend einen guten Überblick.

Die Entscheidung, welche Art von System um- und eingesetzt wird, wurde anhand folgender Anforderungen getroffen:

- Nutzung bestehender Authentifizierungsmechanismen
- Vollautomatische Bewertung der Programmieraufgabe
    - Kompilierung des Quelltextes
    - Verifizierung des geforderten Modells
    - Verifizierung der geforderten Programmlogik
    - Feedback Mechanismus
- Unterstützung mehrerer Programmiersprachen
    - mindestens jedoch Java und Python
- Nutzung vorhandener Programmbibliotheken in den Programmieraufgaben
- Einfache und Übersichtliche Darstellung der Resultate

- Das einzusetzende System muss frei Verfügbar und Erweiterbar sein
- Zugriff über ein Web-Frontend
- Leichte Erweiterbarkeit für weitere
    - Programmiersprachen
    - Lehrmodule

## 2. Systembeschreibung

Das entworfene System besteht aus drei Hauptkomponenten. Das Web-Frontend stellt dabei die eigentliche Website mit allen Interaktionsmöglichkeiten dar, wohingegen das Backend die Verwaltung der angebotenen Inhalte, als auch die Überprüfung der Programmieraufgaben übernimmt. Die dritte und letzte Komponente stellt das Stud.IP als externer Anbieter von Kurs- und Teilnehmerinformationen dar, welches darüber hinaus als Authentifizierungsanbieter dient. Abbildung 1 zeigt die beschriebenen Systemkomponenten, sowie deren Kommunikationsschnittstellen.

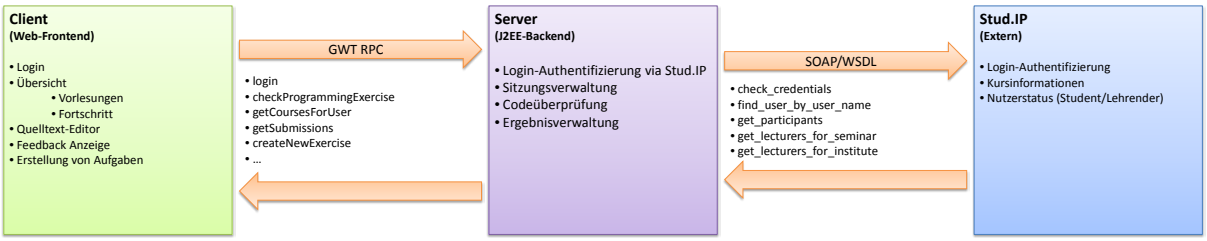


**Abbildung 1: Überblick über Systemkomponenten**

Als Basis für das gesamte System dient das Google Web Toolkit „GWT“ Framework. GWT ermöglicht die Entwicklung von Web-basierten Client-Server Anwendungen. Dabei werden der Server-seitige, als auch der Client-seitige Code komplett in Java erstellt. Der Client-seitige Code wird dann vom GWT-Compiler von Java nach JavaScript übersetzt. Über selbstdefinierte Remote Procedure Calls (RPC) können die Client- und Server-Instanzen miteinander kommunizieren. Weiterführende Informationen finden sich unter [2].

### Web-Frontend

Das Web-Frontend stellt wie bereits beschrieben die Interaktion mit dem Nutzer her. Nach der Authentifizierung des Nutzers (vgl. Abschnitt Authentifizierung) stehen dem Nutzer in Abhängigkeit seines Status unterschiedliche Optionen zur Verfügung. Nutzer haben grundsätzlich die Möglichkeit alle öffentlichen Inhalte abzurufen. Handelt es sich beim angemeldeten Nutzer um ein Stud.IP Konto können ebenfalls Inhalte zur Veranstaltungen abgerufen werden bei denen sich der Nutzer vorher über das Stud.IP als Teilnehmer eingetragen hat. Ist der Nutzer darüber hinaus noch Dozent des Instituts hat dieser entsprechende Editiermöglichkeiten, wie das Hinzufügen neuer Programmieraufgaben. Desweiteren erhalten Dozenten die Möglichkeit die Fortschritte der Kursteilnehmer zu überwachen, als auch die getätigten Quelltextabgaben ansehen.

Abbildung 2 zeigt Beispielhaft die Fortschritte der einzelnen Teilnehmer. Die Veranstaltungsansicht (links) gibt einen Überblick über alle gestellten Aufgaben und welche Aufgabe von welchem Teilnehmer bereits absolviert wurde. Die Aufgabenansicht gibt detaillierte Informationen über die getätigte Quelltextabgabe. Angezeigt werden Bearbeitungsbeginn (Spalte ‚Started‘) und –ende (Spalte ‚Submission‘) der Aufgabe, sowie Anzahl der benötigten Quelltextüberprüfungen (Spalte ‚Runs‘). Über die Schaltfläche „Show“ können die Quelltextabgaben betrachtet werden. Die durchnummerierten Spalten „0“, „1“, usw. entsprechen den der Programmieraufgabe zugrundeliegenden Quelltextdateien. Im gezeigten Beispiel gab es demnach zwei zu Implementierende Klassen. Der jeweilige Eintrag der Spalten ist eine Analyse der jeweiligen Quelltextabgaben zu allen anderen und spiegelt die „Ähnlichkeit“ von Abgaben wieder. Dabei symbolisiert die erste Zahl eine

Gruppe von ähnlichen Abgaben und die zweite die Größe der jeweiligen Gruppe. Die Abgabe „1“ von Teilnehmer drei wurde somit von fünf weiteren, also insgesamt sechs Teilnehmern abgegeben. Um die Ähnlichkeit von Quelltextabgaben zu beurteilen wurde die Editierdistanz bzw. Levenshtein-Distanz jeder Abgabe mit allen anderen Abgaben ermittelt. Liegt die ermittelte Distanz unter einem festgelegtem Schwellwert werden die Abgaben einer Gruppe zugeordnet.

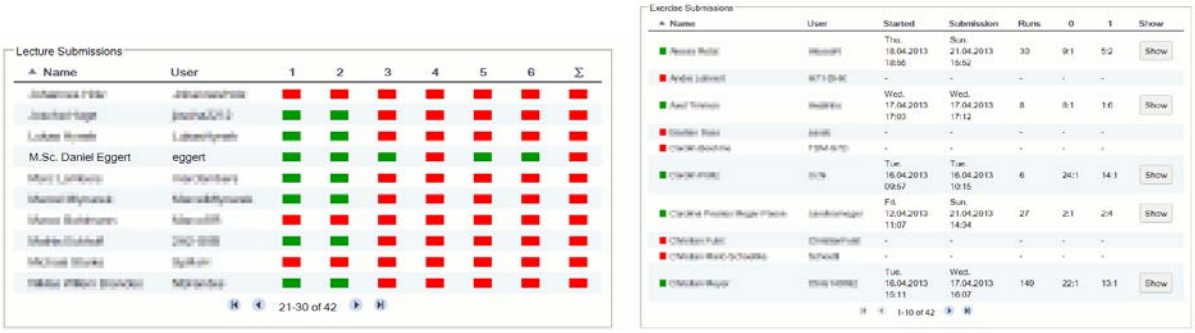


**Abbildung 2: Abgabenübersicht für eine komplette Veranstaltung (links) und für eine einzelne Programmieraufgabe (rechts)**

## Backend

Die Server-Komponente ist wie bereits beschrieben für die Datenhaltung als auch die Überprüfung der Programmieraufgaben zuständig. Die Datenhaltung ist über eine lokale Datenbank realisiert. Um ein einfaches Deploying (Verteilung) des Systems zu gewährleisten wurde für die Datenbank auf ein eingebettetes Datenbanksystem in Form von HSQLBD zurückgegriffen. Die Datenbank ist komplett in das Systemeingebettet und somit nach außen nicht sichtbar.

Neben der eigenen Datenhaltung kommuniziert die Server-Komponente noch mit dem Stud.IP Service um Vorlesungs- und Teilnehmerinformationen abzurufen. Die Kommunikation ist über die vom Stud.IP bereitgestellte SOAP Schnittstelle, welche in [3] dokumentiert ist, realisiert. SOAP steht dabei für Simple Object Access Protocol  und ermöglicht den netzwerkbasierten Austausch von Daten zwischen zwei Systemen. Die jeweiligen Schnittstellen sind WSDL (Web Service Description Language), einer Beschreibungssprache für SOAP-basierte Schnittstellen, definiert. Die Nutzung der SOAP Schnittstelle setzt jedoch einen API Schlüssel voraus, welcher erst beim Stud.IP Betreiber beantragt werden muss. Darüber hinaus sind die Zugriffe auf vorher festgelegte IP Adressen beschränkt. Die vom System genutzten Funktionen sind: `check_credentials` zur Authentifizierung der Nutzer, `find_user_by_username` zur Ermittlung des vollständigen Nutzernamens, `get_participants` liefert eine Liste aller eingetragenen Teilnehmer einer Veranstaltung, `get_lecturers_for_seminar` stellt eine Liste der Dozenten einer Veranstaltung bereit, wohingegen `get_lecturers_for_institute` alle Dozenten eines Instituts liefert. Die genannten Funktionen werden dafür genutzt die entsprechenden Ansichten und Interaktionsmöglichkeiten der Nutzer über das Frontends zu unterscheiden.

## Authentifizierung

Für die Nutzerauthentifizierung wurden zwei Varianten implementiert. Die bevorzugte Methode ist die Stud.IP Authentifizierung. Die SOAP Schnittstelle stellt dafür die Funktion `check_credentials` bereit, mit der Nutzername und Passwort auf ihre Gültig überprüft werden.

Neben dem Stud.IP Webservice ist auch eine Authentifizierung via OpenID umgesetzt. OpenID ist dabei ein dezentrales Authentifizierungssystem, welches einem Benutzer erlaubt sich mit einem OpenID-Konto bei beliebigen das System unterstützenden Websites anzumelden. Fast alle großen Webportale haben den Standard umgesetzt und fungieren somit als OpenID-Anbieter. Um Inhalte für externe Teilnehmer zu Verfügung zu stellen integriert das System den beschriebenen OpenID

Standard und unterstützt bisher Google als Anbieter. Weitere Anbieter, wie z.B. Microsoft, Facebook oder Twitter können bei Bedarf mit minimalem Aufwand hinzugefügt werden.

## 3. Bewertung der Programmieraufgaben

Nach der Eingabe des Quelltextes in die zugehörige Frontend Maske wird dieser per RPC an die Server-Instanz zur Überprüfung geschickt. Dafür muss der Quelltext in Abhängigkeit der verwendeten Programmiersprache zunächst kompiliert werden. Etwaige Compiler-Fehler werden dabei als Feedback im Frontend angezeigt. Anschließend wird die Korrektheit mittels bereitgestellter Unit-Tests überprüft. Zurzeit wird lediglich zwischen Korrekt und nicht Korrekt unterschieden, eine feinere Bewertung kann daher nicht angebracht werden. Jedoch ist es vorgesehen, das der Student ein detailliertes Feedback erhält, welcher Testfall nicht erfüllt wurde und warum er nicht erfüllt wurde. Somit hat der Student Gelegenheit seine Lösung zu korrigieren und erneut zu übermitteln. Die jeweils aktuellste Fassung des übermittelten Quelltextes wird neben weiteren Statistiken in der lokalen Datenbank hinterlegt.

**Modell-Analyse mittels Java-Reflections**

Je nach Aufgabentyp hat sich dabei die Aufteilung in Modell Tests und Logik Tests bewährt. Modell Tests sind dabei über Java Reflections realisiert, wohingegen auf das JUnit Framework im Falle der Logik Tests zurückgegriffen wird.

Das Java Reflection-Modell (häufig als Introspektion bezeichnet) ermöglicht die Analyse und auch Modifikation von Klassen und Objekten, die sich zur Laufzeit im Speicher der Java Virtual Machine JVM befinden. Das Reflection-Modell erlaubt es Felder, Methoden und Konstruktoren von Klassen abzufragen, was sich dazu nutzen lässt die korrekte Umsetzung eines geforderten Modells zu überprüfen. Der häufigste Aufgabentyp im Rahmen der Programmierausbildung ist die Implementierung eines vorgegebenen Modells, welches i.d.R. als UML Klassendiagramm dargestellt oder als reiner Text beschrieben ist.

Die Überprüfung ob der eingereichte Code dem vorgegebenen Modell genügt soll am folgenden Beispiel gezeigt werden. Angenommen es soll folgendes Datenmodell umgesetzt werden:

- *Straßen und Schienenbahnen sind Verkehrswege*
- *Jeder Verkehrsweg besitzt einen Namen*
- *Eine Straße hat eine bestimmte Breite*
- *Schienenbahnen sind „icetauglich“ oder nicht*

Um das implementierte Modell auf Korrektheit zu überprüfen, sind folgende Tests notwendig:

1. Es existiert eine Klasse Schienenbahn, Straße als auch Verkehrsweg
2. Die Klassen Schienenbahn und Straße erben von Verkehrsweg
3. Die Klasse Verkehrsweg besitzt ein Feld mit dem Bezeichner *name* vom Typ String
4. Die Klasse Straße besitzt ein Feld mit dem Bezeichner *breite* vom Type double oder float
5. Die Klasse Schienenbahn besitzt ein Feld mit dem Bezeichner *icetauglich* vom Typ boolean

Überprüfung (1) ist hinfällig, da alle Klassen immer als Code-Stub vorgegeben werden müssen. Um die Vererbungsbedingung (2) zu überprüfen muss zunächst auf die generische Reflections Klasse `Class<?>` zugegriffen werden, dies wird über einen statischen Zugriff auf das Feld `class` der entsprechende Klasse realisiert. Anschließend kann über die Methode `getSuperclass` die Elternklasse und damit die geforderte Vererbung überprüft werden. Listing 1 zeigt die zugehörige Implementierung.

```
Class<Strasse> cl = Strasse.class;
assertTrue("Eine Strasse muss ein Verkehrsweg sein!",
                    cl.getSuperclass().equals(Verkehrsweg.class));
```

**Listing 1: Überprüfung der Vererbungshierarchie mittels Java Reflections**

Neben der Vererbungshierarchie können alle relevanten Modellparameter auf diese Weise überprüft werden. Beispielsweise lässt sich mit `getDeclaredField` auf die Felder einer Klasse über dessen Name zugreifen. Existiert kein Feld mit dem gegebenen Namen wird eine entsprechende `Exception` geworfen. Abschließend lässt sich der Typ des gefundenen Feldes ebenfalls mit `getType` überprüfen. Listing 2 zeigt dies am Beispiel der Bedingung (3), wobei Bedingungen (4) und (5) analog zu überprüfen sind.

```
boolean ex = false;
Class<Verkehrsweg> cl = Verkehrsweg.class;
try {
      Field f = cl.getDeclaredField("name");
      assertTrue("Das Feld Verkehrsweg.name muss vom Typ String sein!",
                          f.getType().equals(String.class));
} catch (NoSuchFieldException e) {
      ex = true;
}

assertFalse("Die Klasse Verkehrsweg muss ein Feld \"name\" besitzen!", ex);
```

**Listing 2: Überprüfung der Klassenfelder mittels Java Reflections**

## 4. Sicherheit

Da das gezeigte System eine Authentifizierung voraussetzt, gilt es die Übertragung der eingegebenen Zugangsdaten entsprechend zu sichern. Grundsätzlich sollte hier auf eine SSL Verschlüsselung via HTTPS zurückgegriffen werden. Ist dies aus technischen Gründen nicht möglich, unterstützt das System auch eine eigene Verschlüsselung der Nutzerdaten. Diese basiert jedoch lediglich auf dem symmetrischen Verschlüsselungsverfahren „Triple-DES“. Da symmetrische Verfahren als leicht Angreifbar gelten sollte diese Variante nur im Ausnahmefall genutzt werden.

Da die Bewertung der Programmierübung darauf beruht den studentischen Quelltext serverseitig zu kompilieren, als auch auszuführen ist eine Sicherung des Servers vor böswilligem oder schlicht falschem Code unerlässlich. Je nach Programmiersprache kommen hier unterschiedliche Konzepte zum Einsatz. Grundsätzlich gilt zunächst, dass eine Codeüberprüfung nur als gültiger Benutzer möglich ist.

Allgemein stellt ein Sandbox-Ansatz, wie er auch in vielen existierenden Systemen wie [4], [5] und [6] Verwendung findet, wohl eine der sichersten Varianten dar. Der vom vorgestellten System verwendete Sandbox-Ansatz nutzt die Java `SecurityManager` Schnittstelle, um die Rechte des zu prüfenden Codes stark einzuschränken. Um die Ausführung der restlichen Server- und Testklassen nicht zu beeinträchtigen ist der `SecurityManager` ausschließlich auf die zu testenden Klassen limitiert. Der implementierte `SecurityManager` verhindert jedweden Zugriff auf bspw. externe Prozesse, das Dateisystem, Internet-Sockets, Classloader, etc.

Als weiteres Merkmal lässt sich die maximale Ausführungsdauer des Testcodes limitieren. So werden versehentliche oder absichtliche Endlosschleifen verhindert. Darüber hinaus führt jede `Exception` im Testablauf zu einem Automatischem Abbruch des Tests und einem entsprechenden Feedback an den einreichenden Nutzer.

Im Falle der ebenfalls vom System unterstützten Programmiersprache Python kann der beschriebene Sandbox-Ansatz nur beschränkt verwendet werden. Hierbei wird der übermittelte Quelltext auf potentiell Gefährliche Fragmente hin überprüft und entsprechend bereinigt. Konkret werden dabei sämtliche Import Anweisungen verworfen und nur die für die Lösung der Aufgabe relevanten Imports wieder hinzugefügt. Darüber hinaus werden sämtliche Anweisungen, die einen Dateisystem- oder Socketzugriff realisieren herausgefiltert. Die Analyse des Python-Quelltextes basiert dabei auf Regulären Ausdrücken, ähnlich wie in [7] beschrieben.

## 5. Zusammenfassung und Ausblick

Es wurde ein System zur automatischen Überprüfung von Programmieraufgaben vorgestellt. Realisiert wurde es als Web-basierte Client-Server-Anwendung mit dem Framework Google Web Toolkit. Das System ist über eine SOAP Schnittstelle an den Stud.IP Service angebunden und nutz dessen Nutzerverwaltung, sowie die Kurs- und Teilnehmerinformationen. Darüber hinaus wurde auf die Modellüberprüfung mittels Java Reflections eingegangen. Abschließend wurden genutzte Sicherheitsmechanismen wie Sandboxing, Quelltextanalyse via Regulären Ausdrücken und Verschlüsselung beschrieben.

Das gezeigte System wird bereits im Lehrbetrieb eingesetzt, nichtsdestotrotz unterliegt es einer konstanten Weiterentwicklung. Geplant sind beispielsweise die Integration weiterer OpenID Anbieter zur Nutzerauthentifizierung. Darüber hinaus wird zurzeit untersucht inwiefern das System oder Teile davon auf Cloudservices wie Googles AppEngine oder Amazons AWS migriert werden kann um eine bessere Skalierbarkeit zu gewährleisten. Die Einführung einer Bewertungsskala über richtig und falsch hinaus wird ebenfalls geprüft.

## 6. Literaturverzeichnis


[1] P. Ihantola, T. Ahoniemi, V. Karavirta und O. Seppälä, „Review of Recent Systems for Automatic Assessment of Programming Assignments," in s *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli, Finland, 2010.

[2] Google Inc., „GWT Project," Google Inc., [Online]. Available: http://www.gwtproject.org/. [Zugriff am 25 09 2013].

[3] Stud.IP Open-Source-Projekt, „NuSOAP: Stud.IP Webservice," [Online]. Available: https://elearning.uni-hannover.de/soap.php. [Zugriff am 27 09 2013].

[4] B. Merry, „Using a linux security module for contest security," in s *Olympiads in Informatics 3*, 2009, pp. 67-73.

[5] M. Amelung, P. Forbrig und D. Rösner, „Towards generic and flexible web services for e-assessment," *SIGCSE Bull.*, pp. 219-224, Juni 2008.

[6] E. a. T. M. A. a. R. J. a. C. F. a. R. S. Gutiérrez, „A new Moodle module supporting automatic verification of VHDL-based assignments," *Comput. Educ.*, pp. 562-577, Februar 2010.

[7] M.-Y. Chen, et.al., „Design and applications of an algorithm benchmark system in a computational problem solving environment," in s *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education*, Bologna, Italy, 2006.