eTutor

Modularchitektur und Moduldokumentation

JDBC

Christian Eichinger

Version 1.3
Stand März 2006

INSTITUT FÜR WIRTSCHAFTSINFORMATIK
dke
Data & Knowledge Engineering

Versionsverlauf:

Version	***Autor***	***Datum***	***Änderungen***
1.0	ce	28.2.2006	Dokumentation Analysekomponente
1.1	ce	01.03.2006	Dokumentation Bewertungskomponente
1.2	ce	02.03.2006	Dokumentation Feedback-Komponente
1.3	ce	07.03.2006	Dokumentation Architektur und Konfigurationsmöglichkeiten

Inhaltsverzeichnis:

1. Einleitung

Das eTutor Modul JDBC ermöglicht Studenten, Java Programme mit JDBC Code über das eTutor-System abzugeben und die Ergebnisse der Programmverarbeitung korrigieren zu lassen. Die Korrektur der Ergebnisse erfolgt dabei an Hand der Verarbeitungsergebnisse des Studentenprogramms und nicht auf Grund semantischer Programmanalysen. Neben Analyse-, Bewertungs- und Feedback-Mechanismen stellt das JDBC Modul ebenfalls Administrationsfunktionalitäten zur Erfassung und Verwaltung von JDBC Beispielen zur Verfügung.

Da die Prüfung der Studentenabgaben ergebnisorientiert funktioniert, d.h. das Endergebnis der Ausführung der Abgabe wird mit einer vordefinierten Musterlösung verglichen, bedient sich das JDBC Modul auch der Funktionalitäten des Moduls SQL zur Bewertung der Ergebnisse. Zum Verständnis der Ausführungsweise des JDBC Moduls wird daher neben der Lektüre des eTutor Architekturdokuments auch die Durchsicht des SQL Moduls empfohlen. Die Beschreibung des Feasibility-Prototypen des JDBC-Moduls kann auch in der Diplomarbeit von Frau Anita Hofer (E-Exercises in Data & Knowledge Engineering) nachgelesen werden.

Die JDBC-Modulbeschreibung ist in 3 Bereiche unterteilt. Abschnitt 2 beschreibt das Funktionsprinzip und die Algorithmen des JDBC Moduls. Es wird dabei auf die Analyse-, Bewertungs-, Feedback- und Administrationsfunktionalitäten des JDBC-Moduls eingegangen. Abschnitt 3 beschreibt die Architektur des JDBC-Moduls und dessen Integration in das eTutor System. Abschnitt 4 beschreibt schließlich die Konfigurationsmöglichkeiten des Moduls.

2. Funktionsweise

Der generelle Ablauf bei der Benutzung eines Moduls wird in Abbildung 2.1 dargestellt: Der Benutzer wählt über einen Browser eine konkrete Übungsaufgabe im eTutor-System aus (*showTask.jsp*). Die ausgewählte Aufgabe kann daraufhin auf einer Seite, die mit dem jeweiligen Modul mitgeliefert wird, ausgearbeitet werden (*showEditor.jsp*). Diese Seite wird wie alle übrigen dynamischen Seiten mittels JSP (Java Server Pages) realisiert. Durch die Auswahl der Übungsaufgabe werden an diese Seite Parameter übermittelt, von denen vor allem die folgenden für das Modul relevant sind:

- *typeID*: Dieser Parameter identifiziert das Modul (im vorliegenden Fall das JDBC Modul).
- *exerciseID*: Dieser Parameter dient dazu, die Übungsaufgabe, die am Server gespeichert ist, zu identifizieren.
- *modeID*: Mit diesem Parameter wird festgelegt, ob die Beispielausarbeitung im Übungs- oder im Abgabemodus erfolgt.

Wenn der Benutzer auf der Seite *showEditor.jsp* die Abgabe bestätigt, werden die oben erwähnten Parameter und die ausgearbeitete Lösung an das *CoreManagerServlet* des eTutor-Systems übermittelt. Von dort werden die Daten an das Modul weitergeleitet, das dem Parameter *typeID* zugeordnet ist, und die Methoden für die Analyse, die Bewertung und das Feedback aufgerufen. Diese befinden sich in einer Klasse, die das eTutor-Interface *etutor.core.evaluation.Evaluator* implementiert. Die Ergebnisse werden für den Benutzer letztendlich auf einer Seite *printReport.jsp* dargestellt, die wiederum mit dem jeweiligen Modul mitgeliefert wird.

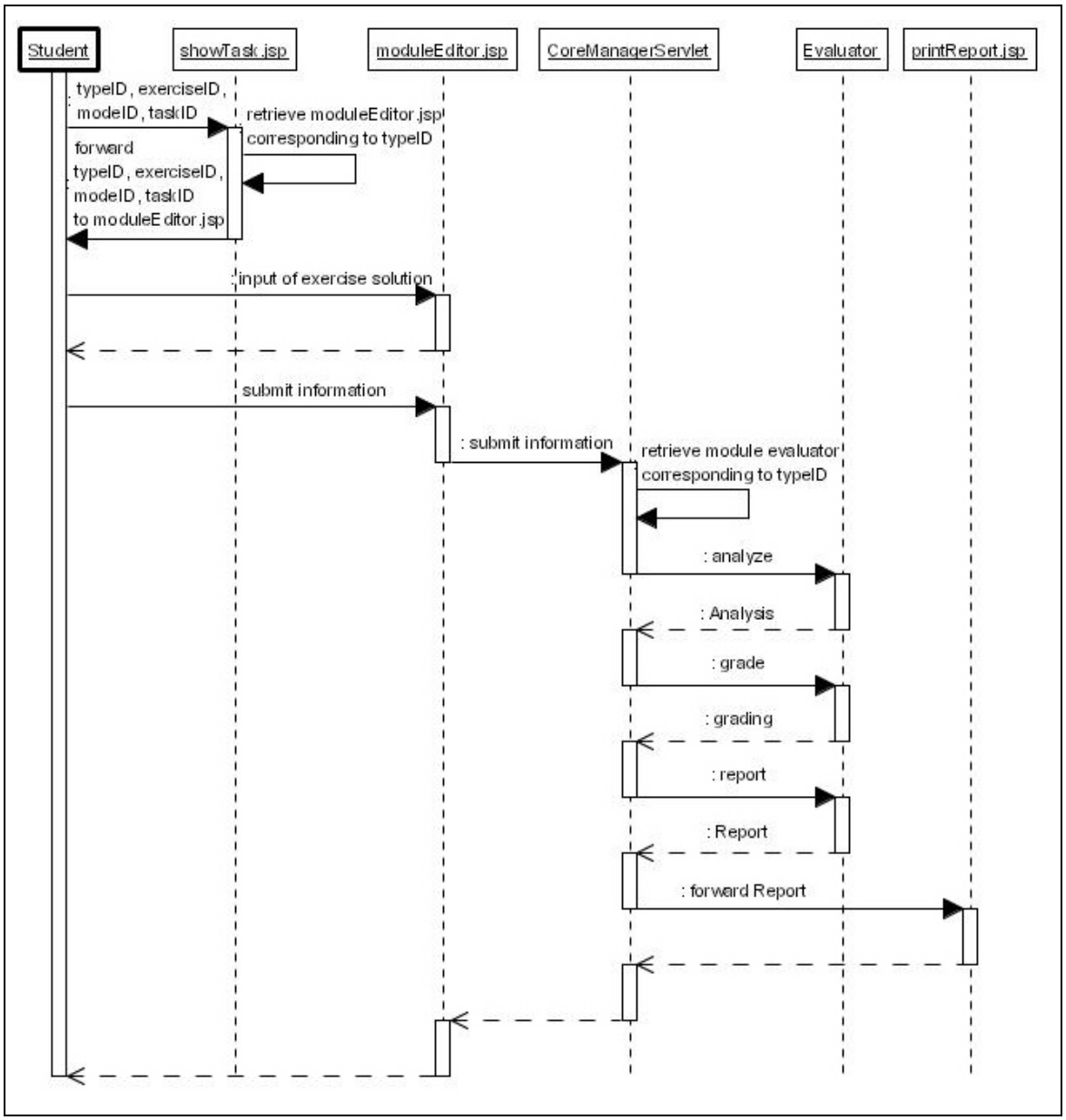

Abbildung 2.1: Aufruf eines Moduls im eTutor

2.1. Analyse

Die Analysefunktion stellt die Kernfunktion der Module dar und dient als Basis für die weiteren Funktionen. Die Funktionalität wird über die folgende Methode, die im eTutor-Interface *etutor.core.evaluation.Evaluator* definiert ist, aufgerufen:

analyze(int exerciseID, int userID, Map passedAttributes, Map passedParameters)

2.1.1. Funktionalität

Die Analyse der Abgabeinformationen des JDBC Moduls folgt dem in Abbildung 2.2 dargestellten Ablauf.

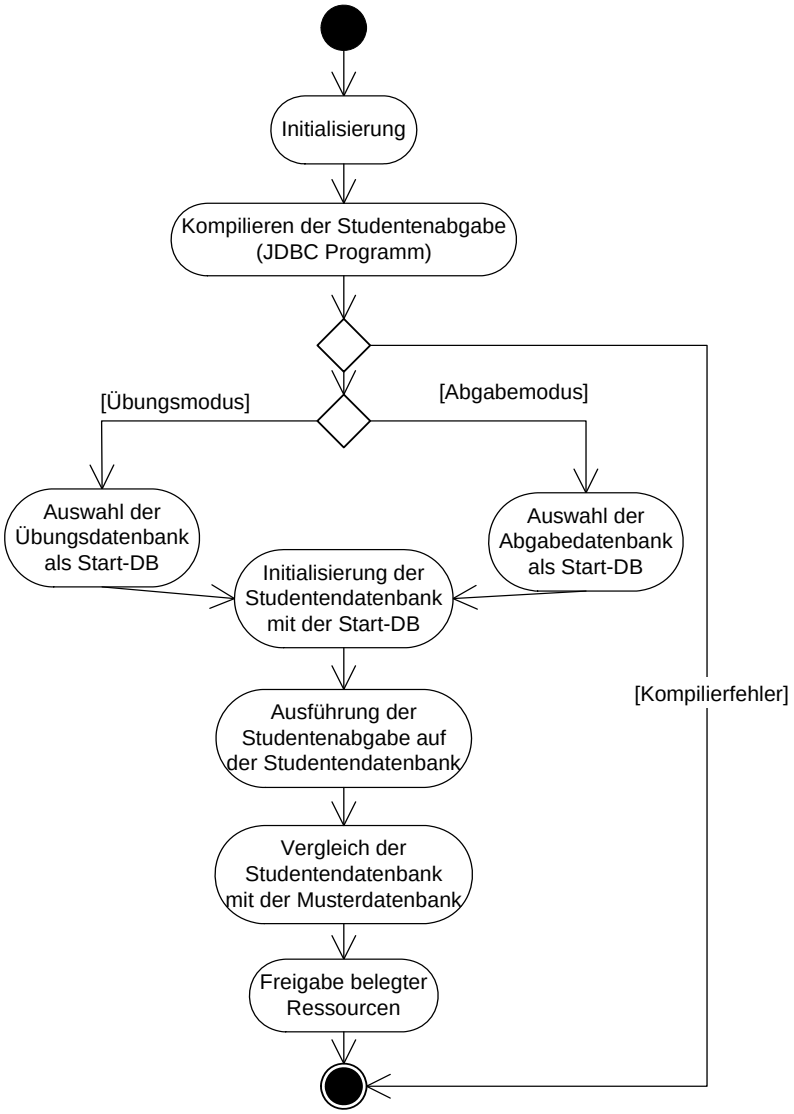

Abbildung 2.2: Sequenz der JDBC-Analyse Funktion

Nachdem der Student über den Editor des JDBC-Moduls sein JDBC-Programm abgegeben hat, wird das Programm an die Analysefunktion weitergegeben. Das Programm wird vom JDBC-Modul kompiliert wobei etwaige Kompilierfehler dem Benutzer angezeigt werden. Anschließend wird die Studentendatenbank, abhängig vom Ausführungsmodus des Moduls, mit Beispieldaten (Start-DB) initialisiert. Auf die so initialisierte Studentendatenbank wird das abgegebene JDBC-Programm ausgeführt. Das Ergebnis der Programmausführung wird mit der Musterlösung des Beispiels verglichen und Abweichungen werden protokolliert. Abschließend werden die belegten DB-Ressourcen wieder freigegeben, d.h. die Studentendatenbank wird wieder geleert. Eine

detaillierte Beschreibung der Abläufe kann den nachfolgenden Funktionsbeschreibungen entnommen werden.

Initialisierung:

In der Initialisierungsphase der JDBC Analyse wird das Rückgabeobjekt etutor.modules.jdbc.analysis.JDBCAnalysis erstellt und initialisiert. Das Analyseobjekt wird in jedem weiteren Bearbeitungsschritt entsprechend dem Erfolg der Ausführung mit Informationen gefüllt. Eine detaillierte Beschreibung der Objektstruktur kann Abschnitt 2.2.3 entnommen werden.

Kompilierung der Studentenabgabe:

Die vom Studenten per Upload bereitgestellte Implementierung der Aufgabenstellung wird in diesem Schritt mittels eines Standard Java-Compilers kompiliert. Zur Realisierung der Upload-Funktionalität wurden Funktionalitäten des Upload-Moduls wiederverwendet, die jedoch für dieses Modul angepasst wurden.

Um eine korrekte Kompilierung der Studentenabgabe zu ermöglichen müssen die abgegebenen Programme folgende Einschränkungen berücksichtigen:

- **Die abgegebene Java-Klasse muss das Interface etutor.modules.jdbc.analysis.JDBCExecuter implementieren:** Dieses Interface enthält die Schnittstelle, über welche das JDBC-Modul Datenbankressourcen an das Studentenprogramm übergibt. Eine Studierendenabgabe, die dieses Interface nicht implementiert, kann nicht ausgeführt werden.
- **Die Studierendenabgabe darf keine Package-Deklarationen enthalten:** Der Java-Compiler setzt voraus, dass Source-Files, welche eine Package-Deklaration enthalten, in einer entsprechenden Verzeichnisstruktur abgelegt sind. Um eine solche Verzeichnisstruktur erstellen zu können, müsste das JDBC-Modul das JDBC-Programm des Studenten parsen. Aus Zeitgründen wurde diese Funktionalität im JDBC-Modul noch nicht realisiert.

Die Kompilierung erfolgt über Methoden der Klasse etutor.modules.jdbc.analysis.JDBCAnalyzer. Die Kompilierung wird über einen Kommandozeilenaufruf an den Java-Compiler realisiert. Die Kompilierung erfolgt somit in einem eigenen Betriebssystem-Prozess. Die Compilermeldungen werden dabei für etwaiges Feedback protokolliert.

Auswahl der Start-DB:

Im JDBC Modul werden zu jedem Beispiel unterschiedliche Datenbestände verwaltet:

- **Übungsdatenbank:** Diese Datenbank enthält jene Daten, auf denen Studierende während der Ausführungsmodi „run“, „check“ und „diagnose“ arbeiten. (Eine detaillierte Beschreibung zu den Ausführungsmodi ist im Architekturdokument des Projektes vorhanden).
- **Abgabedatenbank:** Diese Datenbank enthält jene Daten, auf denen Studierende während des Ausführungsmodus „submit“ arbeiten.

Diese Unterscheidung soll verhindern, dass Studierende durch geschicktes Manipulieren der Ergebnisdaten die algorithmische Aufgabenstellung des Beispiel-Programms umgehen. Durch die Verwendung einer eigenen Abgabedatenbank ist es Studierenden somit nicht mehr möglich, Abgaben zu „fälschen“. Je nach Ausführungsmodus der Aufgabenstellung, wird in diesem Schritt entweder die Übungs- oder die Abgabedatenbank für die weitere Bearbeitung ausgewählt.

Initialisierung der Studentendatenbank:

Jede Übungs- bzw. Abgabedatenbank wird des Weiteren in Beginn- und Enddatenbank untergliedert:

- **Beginndatenbank:** Diese Datenbank enthält den Datenbankzustand der in der Aufgabenstellung des Beispiels beschrieben wird. Je nach Aufgabenstellung wird dieser Zustand der Datenbank durch die Ausführung des Abgabeprogramms verändert.
- **Enddatenbank:** Diese Datenbank enthält jenen Datenbankzustand, der durch eine korrekte Lösung der Aufgabenstellung erreicht wird. Dieser Zustand entspricht somit der Musterlösung.

Um die parallele Auswertung von Abgaben zu ermöglichen ist es notwendig, das jedes Programm eines Studierenden auf einen eigenen Datenbereich ausgeführt wird. Um dies zu erreichen, stehen dem JDBC-Modul eine Reihe von Datenbank-Schemata zur Verfügung, welche beliebig manipuliert werden können. Die Datenbank, und die Anzahl der darauf verfügbaren Schemata, können über die Konfigurationseinstellungen des JDBC-Moduls festgelegt werden.

Jedem Studierenden wird somit ein „eigener“ Datenbankbereich, die *userConnection*, zugewiesen. In diesem Datenbankbereich kann das abgegebene Programm beliebige Datenbankoperationen ausführen. Da jedoch dieser Datenbankbereich ursprünglich leer ist, muss er zuerst mit der Beginndatenbank initialisiert werden. Im Rahmen dieser Initialisierung wird die Beginndatenbank der ausgewählten Start-DB in die zugewiesene *userConnection* kopiert.

Ausführung der Studentenabgabe

Die kompilierte Studentenabgabe wird nun mit der für die Ausführung initialisierten *userConnection* gestartet. Die Übergabe der Datenbankverbindung erfolgt über die Schnittstelle des JDBCExecuter Interfaces:

```
execute(Connection con, PrintStream out)
```

Neben der Datenbankverbindung wird ein java.io.PrintStream Objekt übergeben, welches es den Studierenden ermöglicht, Debugging-Informationen zu protokollieren.

Die Studentenabgabe wird, um größtmögliche Parallelität bei der Ausführung zu ermöglichen, in einem eigenen Java-Thread ausgeführt. Die Ausführung wird dabei vom JDBC-Modul überwacht um etwaigen Ressourcenmissbrauch, Hacker-Angriffen und Programmfehlern entgegenzuwirken. Die Sicherheit der Ausführung wird dabei über einen, speziell für dieses Modul entwickelten Security-Manager (etutor.modules.jdbc.analysis.JDBCSecurityManager) gewährleistet.

Vergleich der Studentendatenbank mit der Musterdatenbank

Nachdem das Programm des/der Studierenden erfolgreich ausgeführt wurde, kann der Zustand der *userConnection* mit der Enddatenbank, also der Musterlösung, verglichen werden. Der Vergleich wird dabei in folgenden Schritten ausgeführt:

(1) **Vergleich des Datenbankschemas:** Es wird verglichen, ob die Lösung des Studierenden dieselbe Datenbankstruktur wie die Musterlösung aufweist. Verglichen werden Tabellen und deren Attribute. Fehlende Tabellen und/oder Attribute sowie nicht übereinstimmende Datentypen von Attributen werden im Analyseergebnis protokolliert.

(2) **Vergleich der Datenbankinhalte:** Es wird verglichen, ob die korrekt vorhandenen Tabellen auch in deren Tupel-Extension übereinstimmen. Bei diesem Vergleich wird die Funktionalität des SQL-Moduls wiederverwendet, d.h. der Tupelvergleich wird an das SQL-Modul deligiert und das Ergebnis in das JDBC-Analyseergebnis integriert.

Das Ergebnis des Vergleichs wird im JDBCAnalysis Objekt vermerkt.

Freigabe belegter Ressourcen

Die zur Ausführung benötigten Ressourcen werden in diesem Schritt wieder freigegeben. Insbesondere wird die vom Studierenden verwendete Datenbankverbindung neu initialisiert, d.h. geleert. Die Datenbankverbindung wird danach in einem Pool verfügbar gemacht und steht somit für weitere Ausführungen zur Verfügung.

Nach erfolgreicher Freigabe der Ressourcen wird die erstellte Analyse an den eTutor-Core zurückgegeben.

2.1.2. Eingabedaten

Neben der Beispielinformationen (*exerciseID*) und dem Modus der Aufgabenauswertung, welcher in der Map passedAttributes enthalten ist, ist bei die Auswertung des JDBC-Moduls der Parameter *userID* besonders wichtig. Dieser wird benötigt, um benutzerspezifische Datenbanken (Studentendatenbank) zu initialisieren und eine eindeutige Zuordnung zu Studenten zu gewährleisten.

Das JDBC-Modul verwaltet zu jedem Beispiel (*exerciseID*) folgende Informationen in einer relationalen Datenbank:

- **Übungsdatenbank:** In dieser Konfiguration werden Verbindungsdaten zu zwei Datenbankschemata aufbewahrt, welche im Übungsmodus der Beispielauswertung benötigt werden: (1) die Beginndatenbank, welche den Datenbankzustand vor der Ausführung des Studentenprogramms enthält und (2) die Enddatenbank, welche den Datenbankzustand zur Musterlösung enthält.

- **Abgabedatenbank:** Die Abgabedatenbank enthält, wie die Übungsdatenbank Verbindungsdaten zu Beginn- und Enddatenbanken, welche im Beurteilungsmodus „submit“ der Beispielauswertung herangezogen werden.

Zur Auswertung von JDBC-Beispielen sind daher vier Datenbankschemata notwendig.

2.1.3. Ausgabedaten

Die Ergebnisse der JDBC Analyse werden in einem Java Objekt der Klasse etutor.jdbc.analysis.JDBCAnalysis zusammengefasst. Die JDBCAnalysis umfasst dabei drei Teilbereiche, die jeweils in eigens dafür vorgesehenen Objekten verwaltet werden:

- **Analyse der Kompilierung:** In einem Objekt der Klasse etutor.jdbc.analysis.CompilationAnalysis verwaltet die JDBCAnalysis Informationen über die erfolgreichen bzw. fehlerhaften Kompilation der Studierenden-Abgabe. Compilermeldung werden in diesem Objekt protokolliert.

- **Analyse der Programmausführung:** Die Rückmeldungen der Programmausführung werden in einem Objekt der Klasse etutor.jdbc.analysis.JDBCRuntimeAnalysis vermerkt. Darunter fallen neben etwaigen Sicherheitsverletzungen oder Programmfehlern auch die Debugging-Ausgaben des abgegebenen Programms.

- **Analyse der Datenbank:** Die Ergebnisse Datenbank-Vergleichs (Tabellen, Attribute und Tupel) werden in einem Objekt der Klasse etutor.jdbc.analysis.DBAnalysis vermerkt.

Das Analyseobjekt wird zu Beginn der Analyseauswertung initialisiert und Schritt für Schritt, entsprechend dem in Abschnitt 2.1.1 beschriebenen Ablauf, mit Informationen gefüllt.

2.1.4. Fehler

Fehler der Analyseauswertung können grundsätzlich in Systemfehler, also Ausnahmesituationen welche die Ausführung des JDBC-Moduls verhindern (z.B. Fehlerhafte Konfigurationsdaten oder Netzwerkprobleme zu einer Datenbank), und Fehler in der Abgabe untergliedert werden. Systemfehler werden grundsätzlich als Java-Exception an den Aufrufer der Analyse zurückgegeben, Fehler der Studentenabgabe werden im JDBCAnalysis Objekt vermerkt um für die Erstellung von Feedbacks herangezogen zu werden. Fehlersituationen können in jedem der einzelnen Verarbeitungsschritt (siehe Abschnitt 2.1.1) des JDBC-Moduls auftreten und werden nachfolgend detaillierter beschrieben.

Initialisierung:

Bei der Initialisierung der Analyse können eine Reihe von Systemfehlern auftreten. So kann es zu Problemen im Verbindungsaufbau mit einer der konfigurierten Datenbankverbindungen (z.B. Beginn- oder Enddatenbank) kommen. Netzwerkfehler werden durch die vom Java RMI Subsystem gelieferten java.rmi.RemoteException angezeigt, Fehler in der Datenbankkonfiguration werden durch den Datenbankserver erkannt und als java.sql.SQLException angezeigt. Abgabefehler sind in der Initialisierungsphase nicht möglich.

Kompilierung der Studentenabgabe:

Bei der Kompilierung der Abgabe in einem eigenen Systemprozess können Systemfehler durch eine fehlerhafte Konfigurationen des Java-Compilers entstehen. Dadurch wird eine Kompilierung der Abgabe unmöglich. Abgabefehler können durch Syntaxfehler des abgegebenen Programms oder durch eine fehlende Implementierung des geforderten etutor.modules.jdbc.analysis.JDBCExecuter Interfaces zustande kommen. Diese Ausnahmesituationen werden in Form von Java-Compiler-Fehlermeldungen an die Studierenden weitergegeben.

Auswahl der Start-DB:

Zu unmittelbaren Fehlersituationen kann es in der Auswahl der Start-DB lediglich durch fehlende Parameter kommen. Wenn der Abgabemodus des Beispiels nicht konfiguriert wurde, wird dies durch eine java.sql.SQLException angezeigt.

Initialisierung der Studentendatenbank:

Bei der Initialisierung der Studentendatenbank kann es zu durch Konfigurationsfehler oder eine fehlerhafte Musterdatenbank zu Systemfehlern kommen. Im Falle eines Fehlers wird dieser vom Datenbanksystem angezeigt.

Ausführung der Studentenabgabe

Bei der Ausführung der Studentenabgabe kann es zu Abgabefehler durch fehlerhafte Programme kommen. Diese werden durch entsprechende Java-Fehlermeldungen bzw. Meldungen des SecurityManagers oder der Systemumgebung angezeigt.

Vergleich der Studentendatenbank mit der Musterdatenbank

Aus dem Vergleich der Studentendatenbank mit der Musterdatenbank werden mögliche Fehlerquellen mit heuristischen Methoden ermittelt. Systemfehler können in diesem Zusammenhang nicht mehr entstehen. Abweichungen der Studentendatenbank mit der Musterdatenbank werden in der etutor.jdbc.analysis.CompilationAnalysis protokolliert.

Freigabe belegter Ressourcen

Sofern die Initialisierung der Datenbankressourcen erfolgreich war, kann die Freigabe ohne weitere Fehler durchgeführt werden.

2.2. Bewertung

2.2.1. Funktionalität

Die Bewertung des JDBC Moduls wird in der Klasse etutor.modules.jdbc.JDBCEvaluator durchgeführt. Die Beurteilung des Abgabe-Programms basiert auf dem Vergleich der Musterdatenbank mit der Ergebnisdatenbank der Programmausführung. Ein Rückschluss auf Programmierfehler innerhalb der Studentenabgabe ist daher nur sehr bedingt möglich. Die Beurteilung des JDBC-Moduls beschränkt sich daher auf die Beurteilung der Richtigkeit der Abgabe – Teilergebnisse werden nicht berücksichtigt. Die Bewertung erfolgt durch 0 und 1, wobei 0 eine Lösung mit Fehlern anzeigt und 1 eine vollkommen korrekte Lösung darstellt.

2.2.2. Eingabedaten

Eingabe-Datum der Bewertung ist das JDBCAnalysis-Objekt der System-Analyse.

2.2.3. Ausgabedaten

Ergebnis der Bewertung ist ein Objekt der Klasse etutor.core.evaluation.DefaultEvaluation. In diesem Objekt wird angezeigt, ob die Ausarbeitung des Studierenden korrekt ist.

2.2.4. Systemfehler

Fehler bei der Bewertung können lediglich durch eine fehlerhafte Initialisierung des DefaultEvaluation Objektes entstehen.

2.3. Feedback

Das Feedback des JDBC Moduls wird, basierend auf dem Analysebericht, schrittweise aufgebaut. Zuerst werden Kompilierungs- und Runtime-Fehlermeldungen angezeigt. Sollten keine dieser Fehler aufgetreten sein, werden Abweichungen zwischen der Musterdatenbank und der Studentendatenbank angezeigt. Sollten keine Fehler aufgetreten sein, wird dies angezeigt. Der Ablauf der Bewertung kann Abbildung 3 entnommen werden.

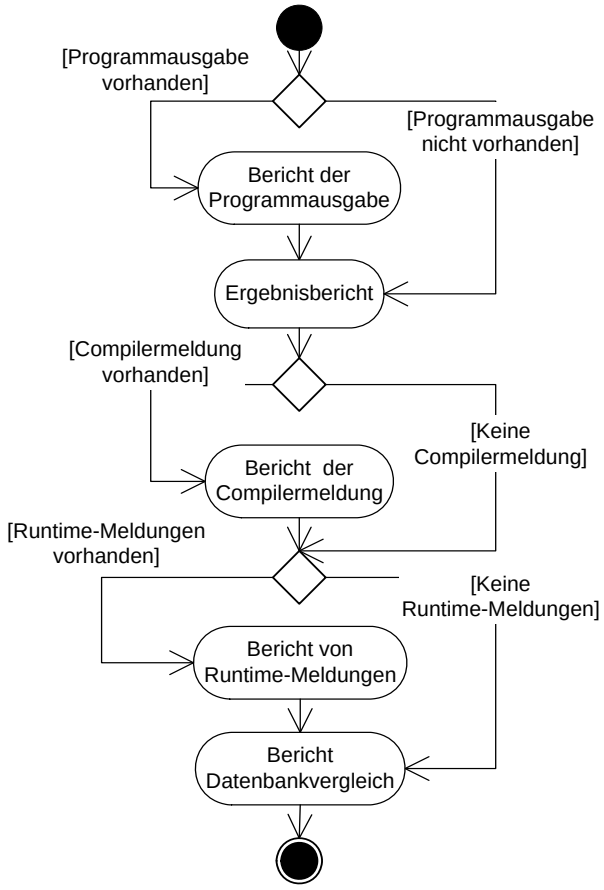

Abbildung 3: Sequenz der JDBC-Berichtsfunktion

2.3.1. Funktionalität

Das Feedback des JDBC Moduls wird in der Klasse etutor.modules.jdbc.report.JDBCReporter durchgeführt. Der Bericht wird dabei in einen Prolog und einen Hauptteil unterteilt. Im Prolog wird das Endergebnis der Programmausführung zusammengefasst (korrekt/nicht korrekt) und etwaige Programmausgaben werden angezeigt. Dieser Berichtssteil ist in jedem Fall vorhanden. Im Hauptteil dargestellte Informationen werden, je nach Verfügbarkeit und etwaiger Diagnosestufe angezeigt. Das Feedback folgt dabei den in Abbildung 3 beschriebenen Schritten.

Bericht der Programmausgabe

Jegliche Ausgaben, welche im Programm des Studierenden auf den übergebenen PrintStream protokolliert werden (siehe auch 2.1.1), werden zu Beginn des Feedbacks angezeigt. Es erfolgt dabei keine Formatierung des angezeigten Textes. Die Programmausgabe wird in jedem Fall, also unabhängig von der Diagnosestufe oder dem Abgabemodus angezeigt.

Ergebnisbericht

Im Ergebnisbericht wird den Studierenden angezeigt, ob die Programmausführung zum gewünschten Ergebnis geführt hat. Es wird eine entsprechende Meldung angezeigt. Die Meldung wird dabei unabhängig von der Diagnosestufe bzw. dem Abgabemodus dargestellt.

Bericht von Compilermeldungen

Jegliche Compilermeldungen werden direkt nach dem Ergebnisbericht angezeigt. Insbesondere Fehler der Kompilierung werden angezeigt. Compilermeldungen werden unabhängig von der Diagnosestufe bzw. dem Abgabemodus dargestellt.

Bericht von Runtime-Meldungen

Fehlermeldungen, welche von der Java-Umgebung während der Ausführung des Abgabe-Programms generiert wird, werden nach Compilerfehlern angezeigt. Diese Fehler werden unabhängig von der Diagnosestufe bzw. dem Abgabemodus dargestellt.

Bericht des Datenbankvergleichs

Sollten Fehler im Datenbankzustand aufgetreten sein werden diese nach etwaigen Runtime-Meldungen angezeigt. Zuerst werden strukturelle Unterschiede zwischen der Musterlösung und der Lösung des Studenten angezeigt, d.h. dass zuerst fehlende/falsche Tabellen und Attribute angezeigt werden. Je nach Diagnosestufe wird dabei nur angegeben, dass Tabellen oder Attribute in der Abgabe fehlen/falsch sind (low), dass eine gewisse Anzahl von Tabellen bzw. Attribute

fehlen (medium) oder welche Tabellen/Attribute fehlen bzw. falsch sind (high). Sollten strukturelle Fehler aufgetreten sein, werden keine inhaltlichen Unterschiede mehr dargestellt, da strukturelle Fehler eine Reihe von Folgefehler generieren würden.

Ist die Struktur der Datenbank korrekt, wird der Tabelleninhalt verglichen. In der Diagnosestufe „low“ wird angezeigt, dass Tupel fehlen bzw. überflüssig sind. In der Diagnosestufe „medium“ wird angezeigt wie viele Tupel überflüssig sind bzw. fehlen und in der Diagnosestufe „high“ wird schließlich angezeigt, welche Tupel fehlen und welche Tupel überflüssig sind.

Im Abgabemodus „check“ bzw. „submit“ werden keine Informationen zum Datenbankvergleich angezeigt.

2.3.2. Eingabedaten

Eingabe-Datum der Bewertung ist das JDBCAnalysis-Objekt der System-Analyse.

2.3.3. Ausgabedateien

Ergebnis des Feedbacks ist ein Objekt der Klasse etutor.modules.jdbc.report.JDBCReport, welcher die Analyse der Ergebnisse in HTML aufbereitet enthält.

2.3.4. Fehler

Wenn gültige Parameter an den JDBCReporter übergeben werden, sind keine Fehler bei der Auswertung zu erwarten.

3. Modularchitektur

3.1. Architektur

Das JDBC Modul besteht aus der Modulkomponente, welche die Schnittstellen zum eTutor Core implementieren, sowie einer Reihe von Datenbanken, welche die Ausführung und Beurteilung von JDBC-Beispielen ermöglichen (siehe Abbildung 4). Das JDBC Modul befindet sich dabei direkt

am eTutor Core Server, da derzeit noch keine RMI-Schnittstellen für das System existieren. Die Datenbanken werden auf dem DB-Server des eTutor Systems verwaltet. Als Datenbank wird derzeit eine Oracle 9i Datenbank verwendet, wobei das JDBC-Modul keine datenbankspezifischen Funktionalitäten verwendet und somit auf eine beliebige SQL 92 kompatible Datenbank portiert werden kann. Neben den Schnittstellenimplementierungen stellt das eTutor-Modul Benutzerschnittstellen in Form von JSP-Seiten zur Verfügung, welche direkt am eTutor Core abgelegt sind.

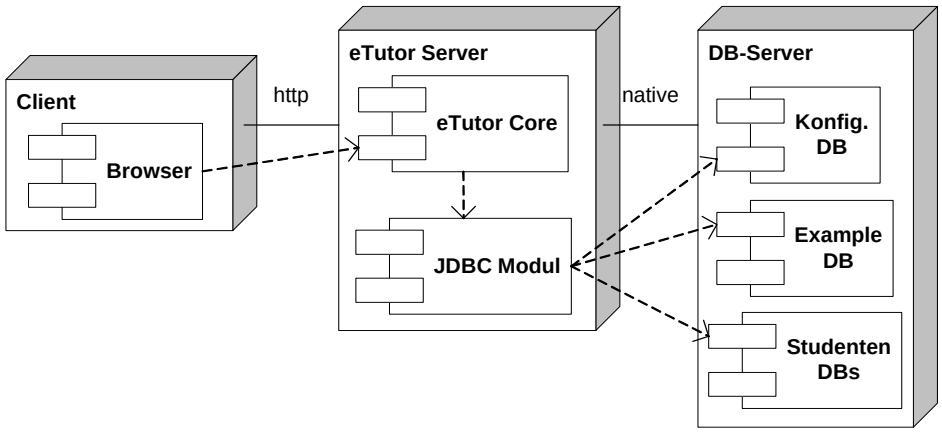

Abbildung 4: Modul-Architektur

3.1.1. Verwendete Klassenbibliotheken (Libraries)

In der Realisierung des JDBC-Moduls wurden die Funktionalitäten und Meta-Informationen, welche über das JDBC-Interface verfügbar sind genutzt. Die verwendete Oracle Datenbank stellt dafür die in Tabelle 3.1 aufgelistete Bibliothek zur Verfügung. Neben den Oracle Funktionalitäten werden auch Librareis zur Verwaltung des Uploads von Studentenabgaben verwendet.

Bibliothek	**Zweck**	**Package**
ojdbc14.jar	Die von Oracle bereitgestellte JDBC Implementierungen für Java Plattformen ab JDK 1.4.	-
cos.jar	Bibliothek zum Parsing von http Multipart Messages in Java. Dieses Package wird für den Upload von Studierenden Abgaben benötigt.	

Tabelle 3.1: Übersicht der verwendeten Klassenbibliotheken

3.1.2. Externe Systeme

Die Analysefunktionalitäten des JDBC Moduls wurden ohne zu Hilfenahme von externen Systemen realisiert.

3.1.3. Package Struktur

Die Packages des JDBC-Moduls wurden entsprechend der Funktionsbereiche Analyse, Bewertung und Feedback strukturiert. Zusätzlich zu den Basisfunktionalitäten wurden Hilfsfunktionen für die Datenbankverwaltung realisiert. Tabelle 3.2 gibt einen überblick über die realisierten Packages und deren Aufgabenbereiche.

Package	**Zweck**
*etutor.modules.jdbc.analysis.**	Dieses Package fasst die Analysefunktionalitäten des JDBC-Moduls zusammen. Dies sind Funktionalitäten zur Kompilierung und Ausführung der Studentenabgaben sowie die Verwaltung der dazu notwendigen Datenbankschemata. Die Verwaltung der Datenbankschemata baut verwendet dabei Hilfsfunktionalitäten des etutor.modules.jdbc.* Moduls.
*etutor.modules.jdbc.report.**	Dieses Package fasst die Feedbackfunktionalitäten des JDBC-Moduls zusammen. Dies sind Funktionalitäten zur Darstellung der Analyseergebnisse entsprechend des gewählten Abgabemodus bzw. der gewählten Diagnosestufe. Die Darstellung der Analyseergebnisse erfolgt dabei in HTML.
*etutor.modules.jdbc.**	Dieses Package Implementiert das vom eTutor Core System geforderte etutor.core.evaluation.Evaluator Interface und stellt somit den Einstiegspunkt in die Funktionalitäten des JDBC-Moduls dar. Von diesem Package aus werden die Analyse und Feedback Funktionalitäten aufgerufen. Da die Bewertungsfunktionalität des JDBC-Moduls nicht sehr komplex ist, wurde diese ebenfalls in diesem Package realisiert. Neben diesen Funktionalitäten stellt dieses Package Hilfsklassen zur Programmkompilation und

	Datenbankmanipulation sowie zur Verwaltung von Datenbankverbindungen zur Verfügung.

Tabelle 3.2: Java-Packages

3.1.4. Datenbank

Das JDBC Modul verwendet die in Abbildung 4 dargestellten Datenbankinhalte zur Konfiguration und Analyse von JDBC Beispielen. Die Datenbankinhalte werden in Tabelle 3.3 näher beschrieben.

Attribut	**Beschreibung**
EXERCISEDB: Diese Tabelle konfiguriert die Zugehörigkeit von Beispielen zu Datenbankschemata, welche für die Ausführung und der Analyse des Beispiels benötigt werden.	
exerciseid	Eindeutige Nummer des Beispiels, welche vom eTutor Core als Beispiel-Identifikation übergeben wird.
begin	Verweis auf das Datenbankschema, welches den Ausgangsdatenbestand für die Ausführung der Studentenlösung enthält. Dieses Schema wird für die Ausführungsmodi „check“, „diagnose“ und „run“ verwendet.
end	Verweis auf das Datenbankschma, welches die Musterlösung des Beispiels enthält. Dieses Schema wird für die Ausführungsmodi „check“, „diagnose“ und „run“ verwendet.
shadow_begin	Verweis auf das Datenbankschema, welches den Ausgangsdatenbestand für die Ausführung der Studentenlösung enthält. Dieses Schema wird für den Ausführungsmodus „submit“ verwendet.
shadow_end	Verweis auf das Datenbankschma, welches die Musterlösung des Beispiels enthält. Dieses Schema wird für den Ausführungsmodus „submit“ verwendet.

CONNECTIONS: Diese Tabelle enthält die JDBC Verbindungsinformationen, welche notwendig sind, um auf die in der Tabelle konfigurierten Datenbankschemata zuzugreifen.	
id	Eindeutige Nummer des Datenbankschemas welches durch diese Konfiguration identifiziert wird.
connect_string	Die für JDBC Treiber benötigte Verbindungsinformation bestehend aus Datebankserver, Port und Datenbank id.
cuser	Name des Datenbank-Benutzers mit Zugriffsrechten auf das konfigurierte Schema.
cpwd	Passwort des cuser für den Zugriff auf das Datenbankschema.
DBUSERS: Enthält die Datenbank-Benutzerkonten, welche für die Ausführung der Studentenabgabe innerhalb der Konfigurationsdatenbank benutzt werden können.	
username	Benutzername, welcher über Rechte zur Ausführung der für JDBC notwendigen Datenbankoperationen (Erstellen und Manipulieren von Tabellen und Views sowie sämtliche DML Operationen) verfügt.
password	Passwort des Datenbankbenutzers.

Tabelle 3.3: Schema der Konfigurationsdatenbank

4. Konfiguration

Die für das JDBC-Modul benötigten Konfigurationseinträge werden in Tabelle 4.1 näher erläutert.

Konfigurationseintrag	**Zweck**
classpath	Dieser Eintrag konfiguriert den für die Kompilierung der Studenten-Abgabe notwendigen Klassenpfad. Dieser enthält die kompilierten, vom eTutor geforderten Interfaces.
lib_directory	Dieser Eintrag enthält Verweise auf Bibliotheken, welche für die Kompilierung der Studenten-Abgabe benötigt wird.
compilation_directory	Dieser Eintrag konfiguriert das Verzeichnis in welchem die Kompilation der Studentenabgabe stattfinden kann. In dieses Verzeichnis werden alle, von der Studentenabgabe benötigten, Dateien kopiert und anschließend kompiliert.
db_connect_string	Dieser Eintrag enthält die Datenbankverbindung, welche die in Abschnitt 3.1.4 beschriebene Datenbank enthält.
db_user	Benutzerkonto, welches zur Verbindung zur benötigt wird.
db_pwd	Benutzerkonto, welches zur Verbindung zur benötigt wird.

Tabelle 4.1: Schema der Konfigurationsdatenbank