Deckblatt

Abgabe der Gruppe 15
(Homepage der Ueb15: http://pcai003.informatik.uni-leipzig.de/~ueb15)

Teil der Aufgabe 3

„Designbeschreibung des Open-Source Projekts WebAssign“

Gruppenmitglieder und aktuelle Rollenverteilung :

Name	**Funktion**	**Email**
Frank Jühling	Projektleiter	mai01dca@studserv.uni-leipzig.de
Markus Scholz	technischer Assistent Verantwortlicher für Recherche	mai01dnn@studserv.uni-leipzig.de
Frank Nowak	Verantwortlicher für Design	mai01chu@studserv.uni-leipzig.de
Philipp Metz	Verantwortlicher für Tests	Metz.Philipp@t-online.de
Stefan Nordt	Verantwortlicher für Implementation	mai01chs@studserv.uni-leipzig.de
Stefan Schmidt	Verantwortlicher für Dokumentation	derstift@web.de

Sie finden die folgenden Dateien im Verzeichnis `Submit` am Sonntag, dem 25.05.03 :

Dateiname	**Teilaufgabe**	**Bearbeitet durch**
`Aufgabe3-Ueb15.pdf` (dieses Dokument)	Designbeschreibung von WebAssign.	Markus Scholz, Frank Jühling
`Pflichtenheft-Ueb15.pdf`	Pflichtenheft	Frank Nowak, Philipp Metz, Stefan Nordt, Stefan Schmidt
`Glossar-Ueb15.pdf`	Glossar	Frank Nowak, Philipp Metz, Stefan Nordt, Stefan Schmidt

Zusammenstellung dieser Ausarbeitung durch Markus Scholz.

1. Allgemeines

1.1 Charakterisierung des Systems

„Das WebAssign-System bietet eine umfassende Plattform für die Durchführung von Übungsveranstaltungen im WWW. Dabei werden Studierende ebenso wie Lehrende unterstützt. Die Erstellung von Aufgaben, das Einsenden von Lösungen, deren Korrektur sowie die Distribution von Ergebnissen erfolgen über das Internet. Den Studierenden werden zeitlich und räumlich flexible und komfortable Arbeitsbedingungen angeboten und den Lehrenden aufwendige administrative Arbeiten und Routinetätigkeiten abgenommen. Die Funktionalität von WebAssign ist auf eine breite Anwendbarkeit und Alltagstauglichkeit ausgelegt.“

Quelle : http://www-pi3.fernuni-hagen.de/WebAssign/ueberblick/uberblick.html

Das WebAssign Projekt das seit 15. März 2001 auch von der FernUni Hagen eingesetzt wird, verfolgt also die gleiche bzw. eine sehr ähnliche Zielstellung wie das von uns angestrebte Übungsbetrieb Projekt und eignet sich deshalb bestens um die Denkprozesse die dahinter stehen zu begreifen und Lösungen zu Problemen, die vielleicht die Entwickler von WebAssign bereits gelöst haben, ähnlich oder vielleicht sogar besser umzusetzen. Natürlich bleibt anzumerken das es sich bei der WebAssign Software eindeutig um ein System mit einer komplexeren Aufgabenstellung handelt das sehr variabel an seinen Einsatzbereich angepasst werden kann. So kann es z.B. den Übungsbetrieb einer gesamten Universität steuern kann aber genauso die Übungen eines einzigen Kurses bzw. Seminars leiten. Die von uns zu entwickelnde Übungsbetriebsoftware ist nur für einen Kurs gedacht und soll außerdem mit deutlich weniger Konfigurationsaufwand und Leistungsanforderungen auskommen. So wird zum Beispiel statt einer (my)SQL-DB als Hauptdatenablage die gesamte Speicherung von Informationen über die Studenten und andere Rollen des Systems im flexiblen XML Format vorgenommen. Aber dazu mehr im Pflichtenheft.

Weiterhin bleibt zu Erwähnen das WebAssign eine weit fortgeschrittene, individuelle Anpassung an die eigenen (Desgin-) Bedürfnisse ermöglicht. Hier wird mit so genannten HTML-Schablonen-Dateien gearbeitet die durch eine Art Parser (natürlich auch ein Servlet) laufen und in welchen entsprechende Variablen (durch „$“ gekennzeichnet) durch entsprechenden (dynamischen) Kontext ersetzt werden, ein System was also ähnlich dem von PHP ist.

1.2 grundlegende Benutzeranforderungen an das System

(nach Rollen geordnet)

Student

- Anmeldemöglichkeit an das System
- gelöste Aufgabe einschicken oder hochladen und erhalt einer (HTML-)Quittung als Bestätigung
- Mitteilungen schreiben/lesen
- für ein/e/en Seminar / Klausur / Kurs an-/abmelden
- Übungsaufgaben herunterladen
- Betrachten der korrigierten Lösungen

(Kurs-)Betreuer

- Einsendungen und Korrekturen von Aufgaben anschauen
- Einsendungs- und Korrekturübersicht über Aufgaben
- Kursressourcen-Verzeichnis betrachten / ändern
- HTML-Schablonen-Dateien können geändert bzw. angeschaut werden.
- Studierende, Korrektoren bearbeiten
- Klausuranmeldungen
- Korrektor Hinweise (z.B. Korrektur Richtlinien) zu Übungsaufgaben zukommen lassen

Korrektoren

- online und offline Korrektur einzelner Aufgaben oder dem gesamten Aufgabenheft (= Übungsserie)
- upload von offline korrigierten Aufgaben
- Benotung der korrigierten Aufgaben
- Übersicht über korrigierte und nicht korrigierte Aufgaben
- Mitteilungen lesen/schreiben

Administrator

- Starten und Stoppen des WebAssign Servers
- Sichern und Archivieren der DB

- Datenmigration aus einer früheren WebAssign Version (Upgrade)
- Konfiguration der Einsendungsserver (die z.B. Multiple Choice automatisch korrigieren können)

Gast

- Zugang zu öffentlichen, ungesicherten Informationen

1.3 Vorraussetzungen des System und Produktumgebung :

Die Vorraussetzungen des Systems gestalten sich zumindest ähnlich wie die des Prototyps, auch wenn sie weitaus umfangreicher sind. Zum Betrieb des WebAssign-Systems ist zunächst ein Web-Server erforderlich. Dieser Server muss in der Lage sein, Java-Servlets anzusteuern. Weiterhin ist ein relationales Datenbankmanagementsystem erforderlich, welches Treiber für JDBC-Verbindungen anbietet. Die Funktionalität des WebAssign-Systems realisiert ein weiterer, in Java implementierter, Server der auf dem Java Developement Kit von Sun basiert. Lauffähig ist das System unter Solaris 2.6 zusammen mit dem Apache Web-Server sowie wahlweise einer Datenbankanbindung an einen Informix Dynamic Server unter Solaris 2.6 bzw. einem MySQL-Datenbankserver unter Solaris oder alternativ unter Linux. Anpassungen an ähnliche Software-Umgebungen sind möglich.

Bei der Hardwareanforderung findet sich in der WebAssign Beschreibung nichts konkretes. Dies ist auch verständlich das da die Anforderungen stark von der Frequentierung des WebAssign-Servers abhängen. Eine generelle Empfehlung läßt sich gegebenenfalls aus dem Lehrgebiet Praktische Informatik III an der FernUni Hagen ziehen. WebAssign läuft dort auf einer Sun Ultra 60 mit zwei UltraSPARC 300 Mhz Prozessoren und 512 MB Hauptspeicher. Die WebAssign-Datenbank befindet sich auf einem separaten MySQL-Server, der auf einer Sun Ultra 10 installiert ist. Mit dieser Konfiguration wird der Übungsbetrieb von Veranstaltungen mehrerer Lehrgebiete der FernUni Hagen realisiert, wobei die Anzahl der studentischen Einsendungen je Veranstaltung und Kurseinheit (14-Tage-Rythmus) von 100 bis zu über 10000 variiert.

2. Produktübersicht

2.1 Stark vereinfachtes Umweltdiagramm des Systems und Überblick über die wichtigsten Funktionen nach dem Einloggen in das System (WWW-Oberfläche)

Überblick über die wichtigsten Funktionen/Seiten nach dem Einloggen in das System (WWW-Oberfläche)
Quelle : Benutzerhandbuch zu WebAssign (ohne Berücksichtigung des Administrators)

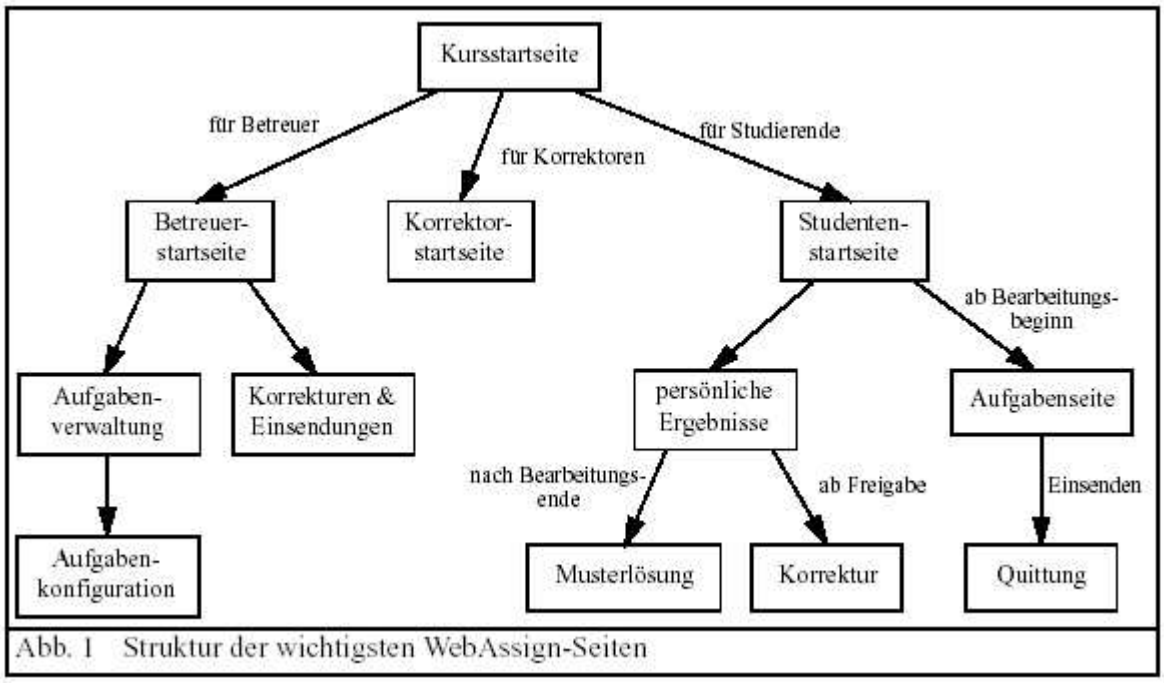
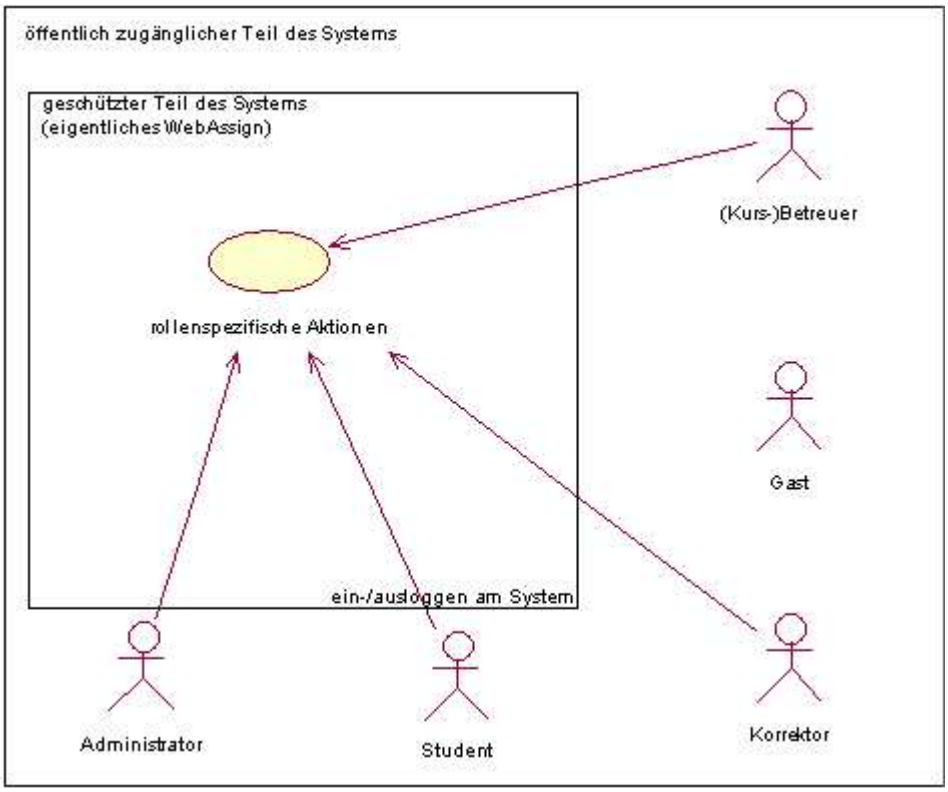


Abb. 1 Struktur der wichtigsten WebAssign-Seiten

Umweltdiagramm :



Anmerkung :

Neben seinen Verwaltungsaufgaben im „inneren“ des Systems hat der Administrator zusätzlich noch viele andere Aufgaben „außerhalb“. So sind spezielle Konfigurationstools (z.B. für die Datenbank von WebAssign) nur ihm unterstellt.

2.2 Anwendungsfälle

Akteuer : alle (außer Gast)

Anwendungsfall:	Nachrichten bearbeiten (schreiben, lesen, löschen)
Beschreibung:	Akteur liest, schreibt oder löscht eine Nachricht.
Ergebnis:	
Vorraussetzung:	Erfolgreiche Anmeldung am System.

Akteur : Student

Anwendungsfall:	**Anmeldung**
Beschreibung:	Anmeldung zu einer WebAssign Sitzung unter Angabe von Matrikelnummer und Passwort.
Ergebnis:	Studentenstartseite erscheint im Browser des Studenten.
Vorraussetzung:	Es existiert ein Eintrag in der Datenbank mit dieser Matrikel/Passwort Kombination.
Anwendungsfall:	**Gelöste Aufgabe hochladen**
Beschreibung:	Student lädt seine Lösung hoch bzw. schickt die ausgefüllte Seite zurück (z.B.Multiple Choice, Lückentext)
Ergebnis:	Es erscheint eine Quittungsseite die das erfolgreiche Hochladen der Lösung bestätigt.
Vorraussetzung:	Erfolgreiche Anmeldung am System.
Anwendungsfall:	**Für ein Seminar, Klausur, Kurs an- oder abmelden**
Beschreibung:	Student meldet sich für ein Seminar, Klausur oder Kurs an oder ab.
Ergebnis:	Bestätigungsseite des System für erfolgreiche Anmeldung. Und Eintrag des Studenten in Kurs/Klausur oder Seminar Liste.
Vorraussetzung:	Erfolgreiche Anmeldung am System.
Anwendungsfall:	**Übungsaufgaben herunterladen oder betrachten (wenn z.B. Multiple Choice oder Lückentext)**
Beschreibung:	Student lädt seine Übungsaufgaben (=Aufgabenheft) herunter oder lässt sie sich anzeigen.
Ergebnis:	Student hast seine Aufgaben auf seiner Festplatte oder sie werden im Browser dargestellt.

Vorraussetzung:	Erfolgreiche Anmeldung am System.
Anwendungsfall:	**Betrachten der korrigierten Lösungen**
Beschreibung:	Student schaut sich die korrigierten Lösungen des Korrektor an.
Ergebnis:	Die korrigierten Lösungen (oder auch Musterlösungen) befinden sich beim Studenten auf der HD oder auch im Browser.
Vorraussetzung:	Erfolgreiche Anmeldung am System, Student hat Lösungen zu diesen Aufgaben hochgeladen bzw. abgegeben und der Korrektor hat diese bereits (vorläufig-) korrigiert.

Akteur : (Kurs-) Betreuer

Anwendungsfall:	**Betrachten der korrigierten Lösungen und nicht korrigierten Lösungen.**
Beschreibung:	Betreuer sieht sich die original oder die korrigierten Lösungen eines Studenten an.
Ergebnis:	Die korrigierten Lösungen (oder auch Musterlösungen) befinden sich beim Betreuer auf der HD oder auch im Browser.
Vorraussetzung:	Erfolgreiche Anmeldung am System und der Student hat Lösungen zu diesen Aufgaben hochgeladen bzw. abgegeben und der Korrektor hat diese bereits (vorläufig-) korrigiert.
Anwendungsfall:	**Betrachten einer Korrekturübersicht bzw. einer Übersicht über die hochgeladenen, gemachten Aufgaben**
Beschreibung:	Betreuer betrachtet Liste der Korrekturen und Abgaben
Ergebnis:	Übersicht über Abgaben und Korrekturen erscheint im Webbrowser des Betreuers.
Vorraussetzung:	Erfolgreiche Anmeldung am System.
Anwendungsfall:	**Betrachten oder Ändern des Kursressourcen Verzeichnisses**
Beschreibung:	Betreuer betrachtet oder ändert Inhalt der Kursressourcen.
Ergebnis:	Betreuer hat sich Überblick über Kursressourcen Verzeichnis gemacht und u.U. neue Daten hinzugefügt.
Vorraussetzung:	Erfolgreiche Anmeldung am System. (Kursressourcen DIR wird standartmässig angelegt)
Anwendungsfall:	**Ändern oder Betrachten der WebAssign Schablonen.**
Beschreibung:	Betreuer ändert oder betrachtet Schablonen Dateien der einzelnen Seiten von WebAssign.
Ergebnis:	Individuelleres Aussehen der WWW-Seiten von WebAssign.
Vorraussetzung:	Erfolgreiche Anmeldung am System oder entsprechend ausreichende Rechte in dem jeweiligen Betriebssystem um auf die Schablonen schreibend/lesend zugreifen zu können.
Anwendungsfall:	**Ändern oder Betrachten der WebAssign Schablonen.**
Beschreibung:	Betreuer ändert oder betrachtet Schablonen Dateien der einzelnen Seiten von WebAssign.
Ergebnis:	Individuelleres Aussehen der WWW-Seiten von WebAssign.
Vorraussetzung:	Erfolgreiche Anmeldung am System oder entsprechend ausreichende Rechte in dem jeweiligen Betriebssystem um auf die Schablonen schreibend/lesend zugreifen zu können.
Anwendungsfall:	**Es sollen alle Korrektoren oder Studenten angezeigt werden.**
Beschreibung:	Betreuer sieht alle Studenten und Korrektoren seine Kurses.
Ergebnis:	Liste der Korrektoren/Studenten wird im Browser des Betreuers angezeigt.
Vorraussetzung:	Erfolgreiche Anmeldung am System.
Anwendungsfall:	**Korrektoren oder Studenten eines Kurses sollen bearbeitet werden.**
Beschreibung:	Betreuter möchte Studenten/Korrektoren seines Kurses anzeigen/anmelden/entfernen/bearbeiten.
Ergebnis:	Bestätigung über Änderung oder Liste der Belegung erscheint im Browser des Betreuers.
Vorraussetzung:	Erfolgreiche Anmeldung am System.
Anwendungsfall:	**Klausuren, Studientage bearbeiten**
Beschreibung:	Betreuer möchte Klausurtermine/Studientage anzeigen lassen oder bearbeiten.
Ergebnis:	Bestätigung über Änderung oder Liste der Termine erscheint im Browser des Betreuers.
Vorraussetzung:	Erfolgreiche Anmeldung am System.
Anwendungsfall:	**Klausuranmeldungen anzeigen lassen oder bearbeiten.**
Beschreibung:	Betreuer möchte Klausuranmeldungen anzeigen lassen oder sie bearbeiten.
Ergebnis:	Bestätigung über Änderung oder Liste der angemeldeten Studenten erscheint im Browser des Betreuers.
Vorraussetzung:	Erfolgreiche Anmeldung am System.
Anwendungsfall:	**Es sollen Hinweise an den Korrektor der Kurse geleitet werden.**
Beschreibung:	Betreuer möchte (z.B. zu den aktuellen Übungsaufgaben) Korrekturhinweise an die Korrektoren

	weitergeben.
Ergebnis:	Betreuer erhält eine Bestätigung und Korrektor(en) erhalten beim Einloggen und Betrachten der Übungsaufgaben der Studenten entsprechende Hinweise.
Vorraussetzung:	Erfolgreiche Anmeldung am System.

Akteur : Korrektor

Anwendungsfall:	**Korrektur einzelner Aufgaben oder des gesamten Übungsheftes (=Übungsserie)**
Beschreibung:	Korrektor möchte einzelne Aufgaben/gesamtes Übungsheft eines Studenten korrigieren. (online oder offline)
Ergebnis:	Korrektor hat die Lösungen eines Studenten auf seiner Festplatte oder im Browser und kann diese korrigieren (über spezielle Subprogramme des WebAssign, welche auch selbst geschrieben werden können)
Vorraussetzung:	Erfolgreiche Anmeldung am System.
Anwendungsfall:	**Offline korrigierte Aufgaben sollen wieder in das System eingespeist werden.**
Beschreibung:	Der Korrektor hat die Aufgabe(n) eines oder mehrerer Studenten kontrolliert und korrigiert und möchte sie zurück ins System stellen.
Ergebnis:	Die korrigierten Aufgaben befinden sich wieder im System. Die Studenten können sich die ihre korrigierten Lösungen anschauen.
Vorraussetzung:	Erfolgreiche Anmeldung am System.
Anwendungsfall:	**Benotung korrigierter Aufgaben**
Beschreibung:	Korrektor möchte die Lösung(en) eines/mehrerer Studenten benoten bzw. ggf. Benotung ändern.
Ergebnis:	Punktezahl des Studenten wird in entsprechende Liste (bzw. DB) übernommen und Student kann darauf zugreifen.
Vorraussetzung:	Erfolgreiche Anmeldung am System. Und Korrektor hat die Lösungen des Studenten bereits korrigiert bzw. betrachtet.
Anwendungsfall:	**Übersicht über die nicht-/korrigierten Aufgaben der Studenten.**
Beschreibung:	Korrektor möchte sich einen Überblick über die bereits korrigierten Lösungen beschaffen.
Ergebnis:	Liste mit allen Studenten in diesem Kurs erscheinen im Browser des Korrektors. Es wird angezeigt für welche Studenten bereits die Aufgaben bereits korrigiert wurden.
Vorraussetzung:	Erfolgreiche Anmeldung am System.

Akteur : Administrator

Anwendungsfall:	**Starten und Stoppen des WebAssign Servers.**
Beschreibung:	Administrator möchte den WebAssign Server stoppen oder starten. Weil z.B. größere Änderungen an der Konfiguration getätigt wurden.
Ergebnis:	WebAssign Server läuft oder läuft nicht.
Vorraussetzung:	Erfolgreiche Anmeldung am System oder entsprechende Rechte für die Ausführung der erforderlichen Programme.
Anwendungsfall:	**Sichern und Archivieren der Datenbank**
Beschreibung:	Der Administrator möchte die WebAssign Datenbank sichern.
Ergebnis:	Archivierte Datenbank.
Vorraussetzung:	Entsprechende Rechte für die Ausführung der erforderlichen Programme.
Anwendungsfall:	**Datenmigration aus einer früheren WebAssign Version**
Beschreibung:	Administrator möchte eine neuere WebAssign Version auf den Rechnern zum Laufen bringen und dabei die alte Datenbank übernehmen.
Ergebnis:	Neue Version mit „alten“ Daten.
Vorraussetzung:	Entsprechende Rechte für die Ausführung der erforderlichen Programme.
Anwendungsfall:	**Es sollen Aufgaben eines bestimmten Typs automatisch durch WebAssign und seine Komponenten automatisch korrigiert werden.**
Beschreibung:	Ein neuer Einsendungsserver muss installiert und konfiguriert werden damit Aufgaben automatisch konfiguriert werden können.
Ergebnis:	Studenten können Aufgaben lösen und erhalten sofort das Ergebnis der Auswertung ihrer Lösung.
Vorraussetzung:	Entsprechende Rechte für die Ausführung der erforderlichen Programme.

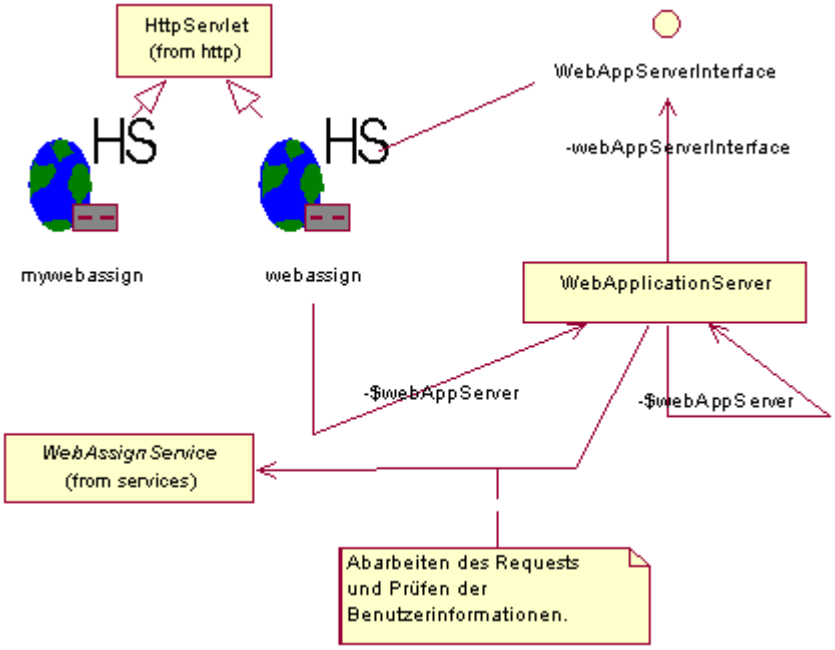
Akteuer : Gast

Anwendungsfall: Ungesicherte Informationen betrachten.	
Beschreibung:	Ein Besucher (ohne Login) möchte sich die Informationen auf der Startseite für das WebAssign anschauen um vielleicht eine groben Überblick über die Funktionalität zu bekommen.
Ergebnis:	Die WebAssign-Startseite erscheint in seinem Browser u.U. kann er außerdem z.B. den Terminplan für die Vorlesung des Seminars o.ä. betrachten.
Vorraussetzung:	-

3. Grundsätzliches Design und Architektur

WebAssign basiert auf einem zentralen Servlet („webassign" – siehe auch 4.) das über den WebServer aufgerufen wird. Dieses „Hauptservlet" leitet den Request weiter an den WebApplicationServer. Dort wird die URL analysiert, der aufgerufene Dienst extrahiert und die Zugangsdaten des Benutzers geprüft. WebAssignService stellt die Basisfunktionalität für alle Service-Klassen zur Verfügung, etwa den Aufbau des Antwort-Datenstroms und die Expansion der darin enthaltenen Variablen (z.B. bei den HTML-Schablonen). Außerdem werden hier Content-type und Content-length gesetzt. Services werden in GET, POST und PUT unterschieden. Unter den Service-Klassen kommt dabei der Klasse „EinsendungsPostService" eine besondere Bedeutung zu. Sie ist dafür verantwortlich das die eingesandten Daten in der Datenbank abgelegt werden, außerdem wird hier die automatische Korrektur einer Aufgabe angestoßen.
Auf der Applikationsebene kooperiert das webassign-Servlet mit anderen Programmmodulen, teils als Dienstgeber, teils als Dienstnehmer. Als *Dienstnehmer* kommuniziert er mit sog. EinsendungsServern. Dies sind Korrekturmodule, in Java oder anderen Sprachen implementiert, die von den Übungsveranstaltungen für ihre speziellen Belange entwickelt und betrieben werden. Ihre physikalische Lokation ist im allgemeinen nicht auf dem Host, auf dem WebAssign betrieben wird. Bei einer solchen Kommunikation „verschickt" der WebAssign-Server eingesendete Lösungen auf Basis von CORBA an einen EinsendungsServer und erhält die fertige automatische Korrektur zurück. Als *Dienstgeber* fungiert der WebAssign-Server z.B. für das Offline-Korrektorkit. Dieses Java-Progamm, das wegen seiner Realiserung als lokaler WebServer auch Korrektur-*Server* genannt wird, installiert ein Korrektor bei sich zu Hause auf dem Home-PC. Infolge entsprechender Benutzereingaben fordert es beim WebAssignServer neue Korrekturen für den jeweiligen Korrektor an bzw. lädt korrigierte Einsendungen herauf. Basis dieser Kommunikation ist http.
Für die lokale Datenhaltung wird eine relationale Datenbank verwendet, die über JDBC angsprochen wird. JDBC ist auch Grundlage des Datenaustauschs mit den zentralen Datenbanken der Hochschulinfrastruktur (HIS), etwa der Beleger-Datenbank. Für die Abfrage von Zugangsberechtigungen dient das LDAP-Protokoll, über das ein Directory-Server angesprochen werden kann. Außerdem bietet WebAssign eine CORBA-Schnittstelle für andere VU-Komponenten, die externe Komponenten befähigt, z.B. Kurse innerhalb von WebAssign anzulegen.

Grundsätzlich lässt sich die Kommunikation der 2 Hauptservlets mit dem WebAppServer (von dem aus alle Aktionen ablaufen) wie folgt darstellen :



HS mywebassign :
Zentrales Servlet von WebAssign zur Weiterleitung und Durchfuehrung von Service-Anfragen. Es handelt sich hier um einen personalisierten Zugriff auf WebAssign.

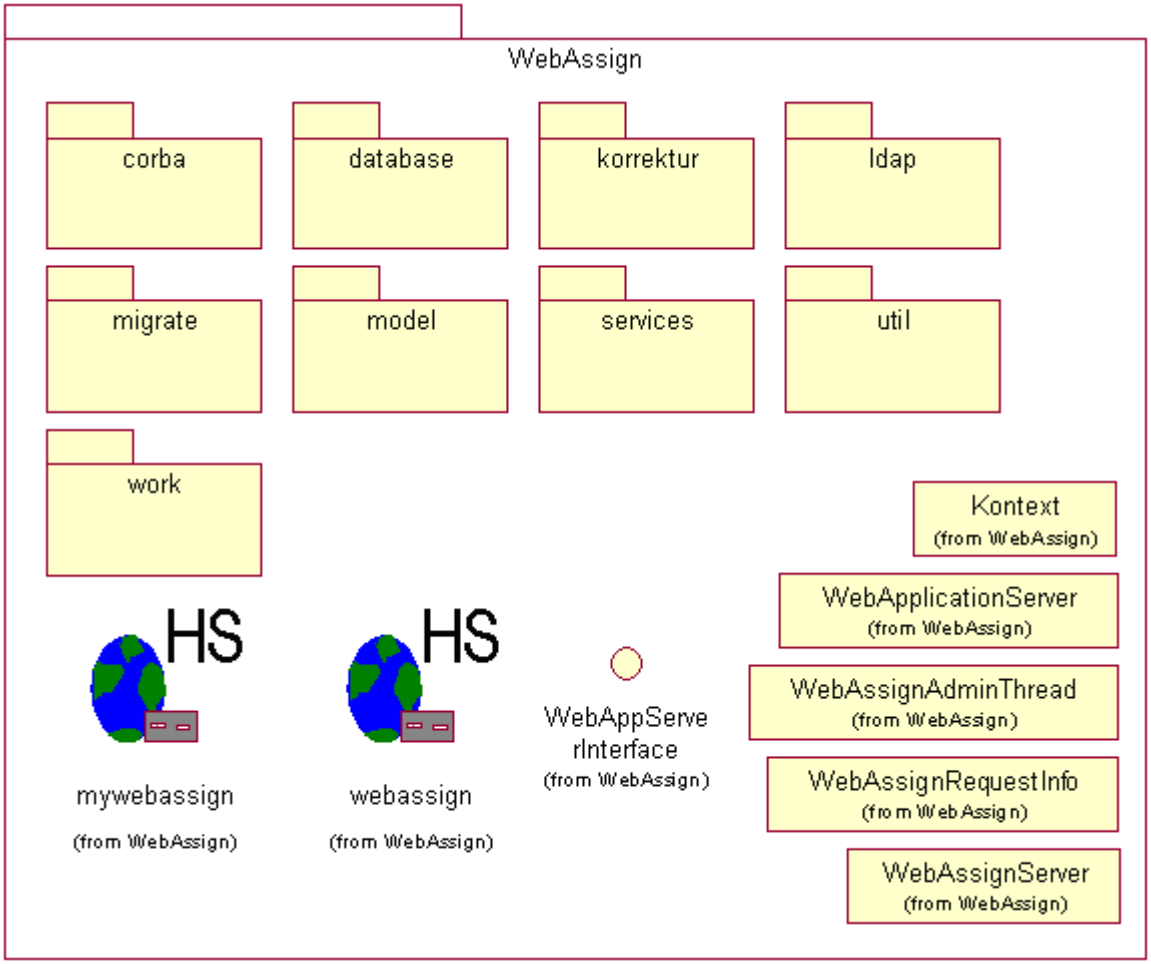
HS webassign :
Zentrales Servlet von WebAssign zur Weiterleitung und Durchfuehrung von Service-Anfragen.

WebApplicationServer :
Ermittelt den angeforderten Service anhand der URL und ruft ihn auf. In der laufenden WebAssign-Anwendung existiert nur eine Instanz von WebApplicationServer. Diese Instanz wird beim erstmaligen Laden des Servlets webassign erzeugt.

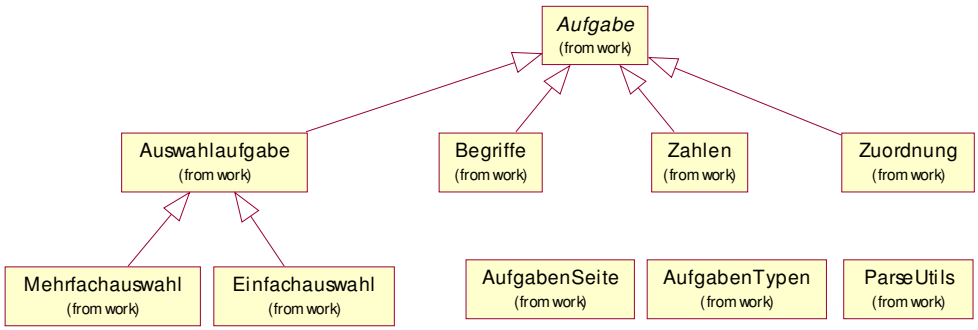
4. Grobe Klassenstruktur des WebAssign Projekts :

Anmerkung : In der Regel sind alle Diagramme bis zur einer Tiefe n=0,1 oder 2 dargestellt. Da sonst die Darstellung zu unübersichtlich werden würde. Außerdem werden nur einige Pakete mit ihren Klassen und Abhängigkeiten dargestellt. (diese sind : WebAssign, WebAssign.services, WebAssign.work, WebAssign.database, WebAssign.korrektur)

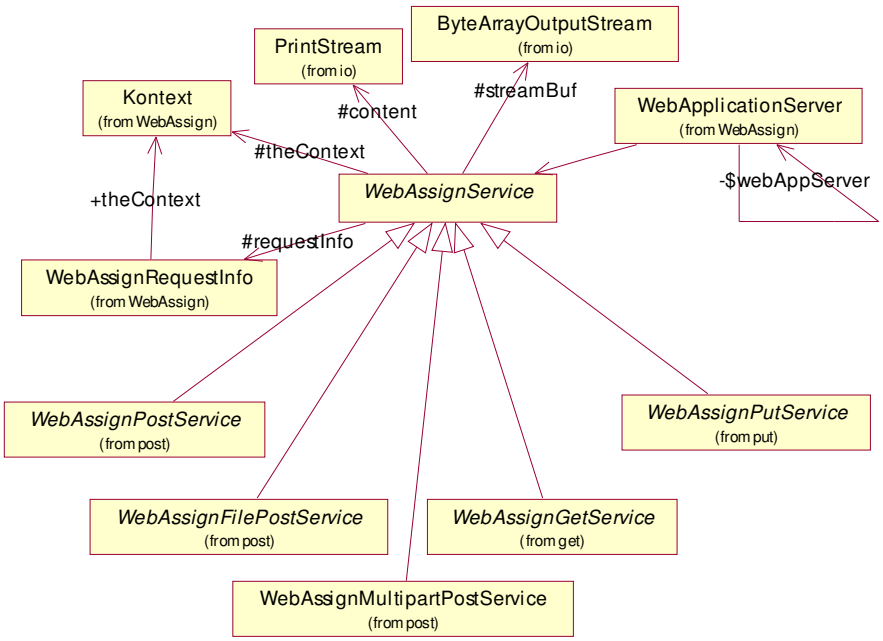
package WebAssign (mit allen Paketen, Klassen und Schnittstellen (Relationen entfernt) :



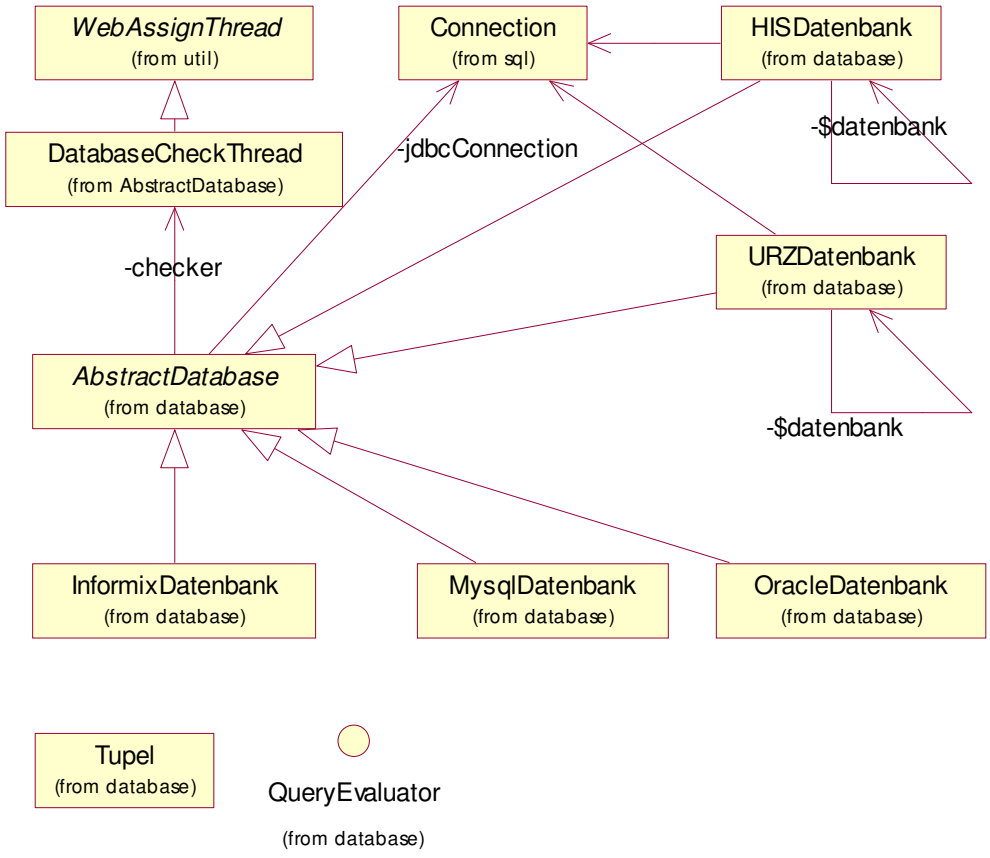
package WebAssign.work :



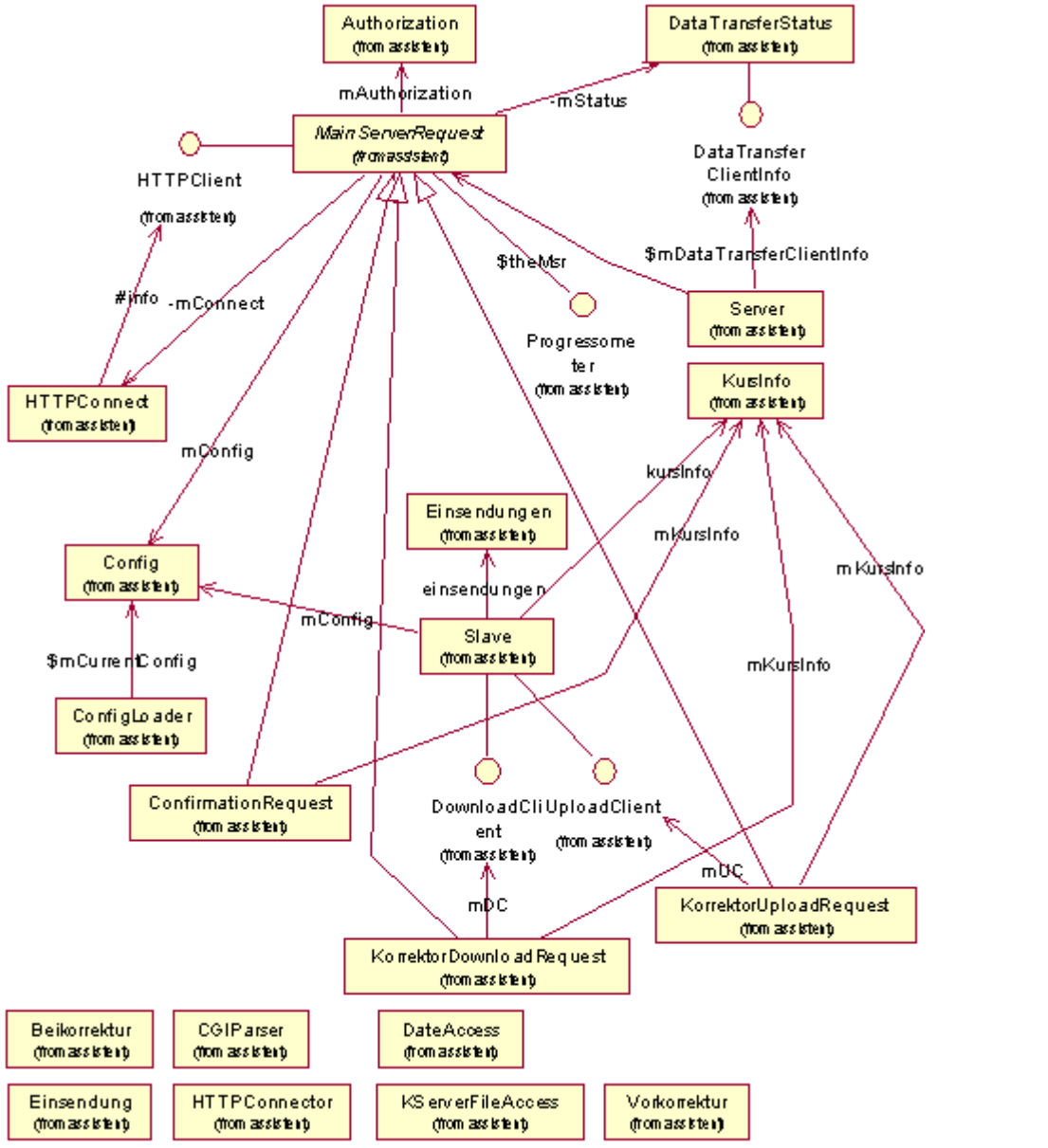
package WebAssign.services :



pachage WebAssign.database :



package Webassign.korrektur.assistent :



package Webassign.korrektur.server :