

# **Entwicklerdokumentation**

**ILIAS Review Plugin**

SWT-Gruppe 04

Auftraggeber: Thordis Kombrink

Weitere: Dr Birgit Demuth, Professor Wollersheim

Tutor: Andy Püschel

16. Januar 2015

Version: 1.0, Status: in Arbeit

# Inhaltsverzeichnis

<b>1</b>	<b>Einführungen und Ziele</b>	<b>4</b>
1.1	Aufgabenstellung . . . . .	4
1.2	Qualitätsziele . . . . .	4
1.3	Stakeholder . . . . .	5
<b>2</b>	<b>Randbedingungen</b>	<b>6</b>
2.1	Konventionen . . . . .	6
<b>3</b>	<b>Kontextabgrenzung</b>	<b>7</b>
3.1	Fachlicher Kontext . . . . .	7
3.2	Externe Schnittstellen . . . . .	8
3.2.1	Review . . . . .	8
3.2.2	Fragepool . . . . .	10
<b>4</b>	<b>Lösungsstrategie und Entwurfsentscheidungen</b>	<b>11</b>
4.1	Fachlicher Kontext . . . . .	11
4.2	Reviewbare Frage-Plugin . . . . .	11
4.2.1	Ordnerstruktur . . . . .	11
4.2.2	Klassenübersicht . . . . .	12
4.3	Review-Plugin . . . . .	13
4.3.1	Ordnerstruktur . . . . .	13
4.3.2	Klassenübersicht . . . . .	13
4.3.3	Besonderheiten der eigenen GUI-Klassen . . . . .	17
4.4	Datenbankstruktur . . . . .	18
<b>5</b>	<b>Bausteinsicht</b>	<b>19</b>
5.1	Entwurfsklassendiagramm . . . . .	19
5.2	Reviewable Question . . . . .	20
5.3	Review . . . . .	21
<b>6</b>	<b>Laufzeitsicht</b>	<b>23</b>
<b>7</b>	<b>Konzepte</b>	<b>26</b>
7.1	Fachliche Strukturen und Modelle . . . . .	26
7.1.1	Analyseklassendiagramm . . . . .	26
7.1.2	Sequenzdiagramm - Review anfordern . . . . .	27
7.1.3	Sequenzdiagramm - Review abschließen . . . . .	28

7.2	Persistenz . . . . .	28
7.3	Benutzeroberfläche . . . . .	29
7.4	Ergonomie . . . . .	30
7.5	Transaktionsbehandlung . . . . .	31
7.6	Sessionbehandlung . . . . .	31
7.7	Sicherheit . . . . .	31
7.8	Plausibilisierung und Validierung . . . . .	31
7.9	Ausnahme-/Fehlerbehandlung . . . . .	31
7.10	Logging, Protokollierung, Tracing . . . . .	32
7.11	Konfigurierbarkeit . . . . .	32
7.12	Internationalisierung . . . . .	32
7.13	Testbarkeit . . . . .	32
7.14	Buildmanagement . . . . .	32
<b>8</b>	<b>Glossar</b>	<b>33</b>

# 1 Einführungen und Ziele

## 1.1 Aufgabenstellung

Aufgabe ist es ein Plugin für das freie, internetbasierte Lernsystem ILIAS<sup>1</sup> zu programmieren, um es um eine Review-Möglichkeit von erstellten Fragen zu erweitern. Die ausführliche Aufgabenstellung, die wir während des ersten Kundentreffens mit Frau Kombrink erhalten haben, liegt im Verzeichnis 'Aufgabenstellung'.

## 1.2 Qualitätsziele

Anforderung	Sehr wichtig	Wichtig	Normal	Irrelevant
Funktionalität	✓			
Benutzerfreundlichkeit	✓			
Erweiterbarkeit		✓		
Zuverlässigkeit		✓		
Korrektheit		✓		
Sicherheit			✓	
Effizienz			✓	
Portierbarkeit				✓

---

<sup>1</sup>[http://www.ilias.de/docu/ilias.php?baseClass=ilrepositorygui&reloadpublic=1&cmd=frameset&ref\\_id=1](http://www.ilias.de/docu/ilias.php?baseClass=ilrepositorygui&reloadpublic=1&cmd=frameset&ref_id=1)

### 1.3 Stakeholder

Name	Rolle	Beschreibung	Intention
Thordis Kombrink	Kundin	Mitarbeiterin an der Fakultät Informatik, zugehörig zum Lehrstuhl Systems-Engineering	-möchte in ihrer Arbeitsgruppe ILIAS nutzen, um Klausuren mit gereviewten Fragen zu erstellen
Dr. Birgit Demuth	Interessentin	Arbeitet am Lehrstuhl Softwaretechnologie	-Interesse an der Umsetzung, um ILIAS unter Umständen an der Fakultät einzusetzen
Prof. Dr. Heinz-Werner Wollersheim	Fachlicher Ansprechpartner	Professor aus Leipzig, der sich mit dem Thema (Fragen für Klausuren reviewen) auseinandersetzt	-würde gerne effektiver mit ILIAS in seiner Arbeitsgruppe arbeiten, dazu fehlt die Reviewmöglichkeit, die wir erstellen

## 2 Randbedingungen

### 2.1 Konventionen

Das Plugin muss die ILIAS-Version 4.4.5, die MySQL Version 5.0.11 und die PHP-Version 5.5.11 unterstützen. Es gelten die im 'ILIAS-Developement-Guide'<sup>1</sup> und in den 'ILIAS Usability and Accessibility Guidelines'<sup>2</sup> festgesetzten Konventionen.

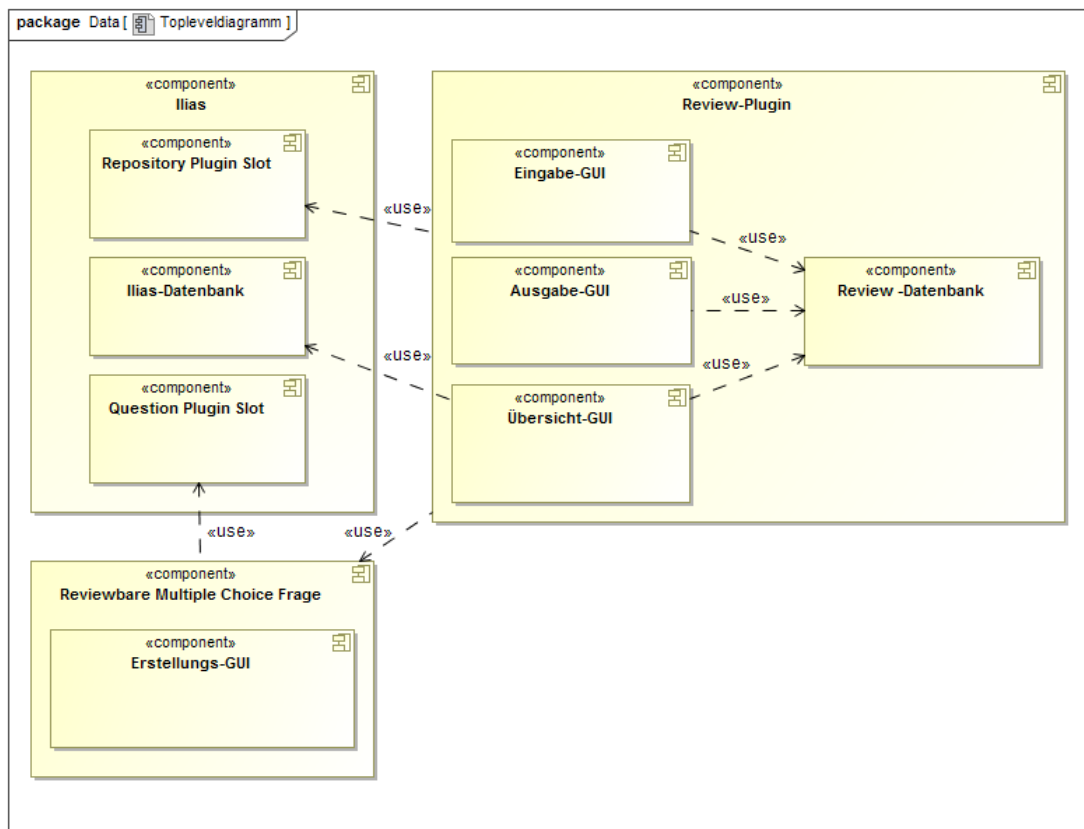
---

<sup>1</sup>[http://www.ilias.de/docu/goto.php?target=lm\\_42&client\\_id=docu](http://www.ilias.de/docu/goto.php?target=lm_42&client_id=docu) (29.10.14)

<sup>2</sup>[http://www.ilias.de/docu/goto\\_docu\\_lm\\_459.html](http://www.ilias.de/docu/goto_docu_lm_459.html) (29.10.14)

## 3 Kontextabgrenzung

### 3.1 Fachlicher Kontext



Um die Review-Funktionalitäten umzusetzen, mussten zwei Plugins für ILIAS entwickelt werden. Zum einen ein Repository-Plugin, das alle Verwaltungsaufgaben für Reviews übernimmt und eine TestQuestionPlugin, das eine reviewbare Frage implementiert.

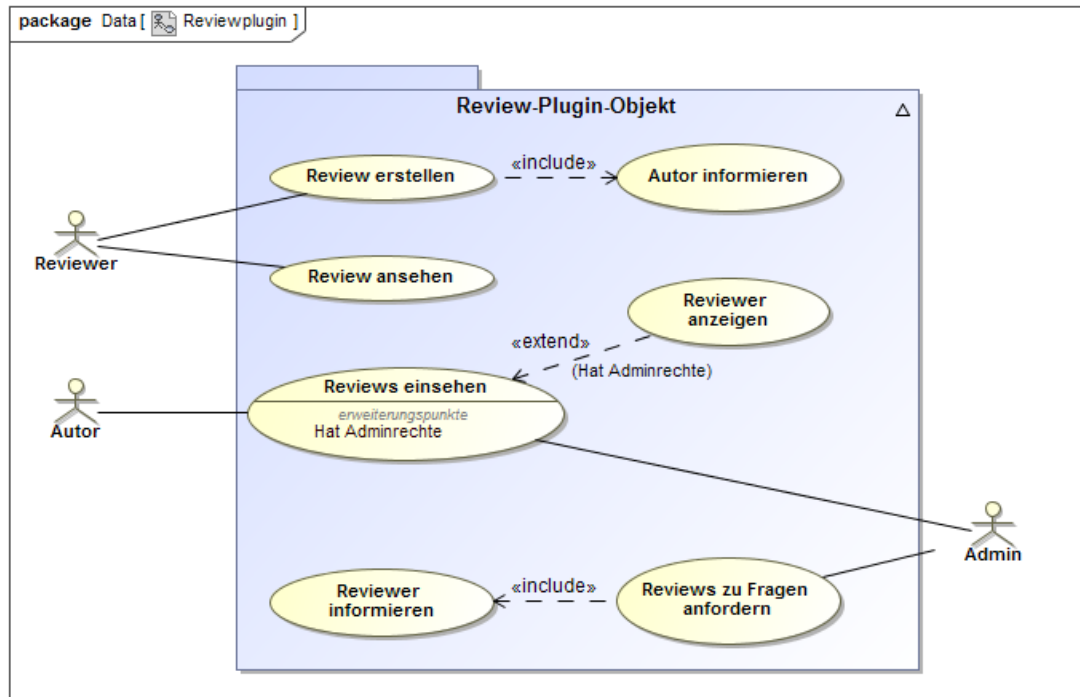
Das Repository-Plugin muss für seine Verwaltungsaufgaben Oberflächen zum Eingeben und Einsehen von Reviews, sowie eine Oberfläche zur Übersicht bereitstellen.

Eine reviewbare Frage braucht, weil sie zusätzliche Daten beinhaltet, eine neue Eingabeoberfläche.

Die beiden Plugins werden an von ILIAS bereitgestellten Schnittstellen angesetzt. Zusätzlich benutzen beide die ILIAS-Datenbank um ihre Daten zu speichern.

## 3.2 Externe Schnittstellen

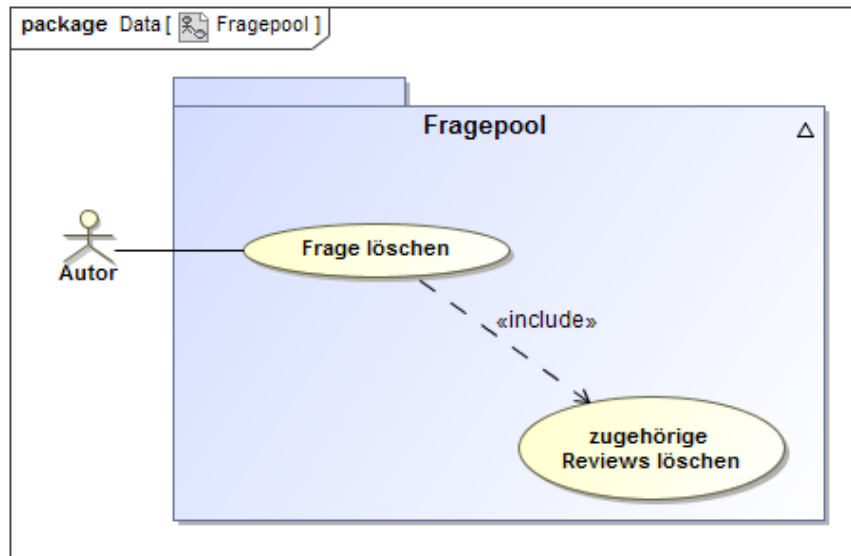
### 3.2.1 Review





Nr.	Anwendungsfall	Nr.	Anfrage	Antwort
1.0	Ein Administrator fordert Reviews	1.1	Es ist noch kein Review vorhanden	Die vom Administrator zur Frage zugeordneten Reviewer werden ermittelt und unbearbeitete Reviews erstellt. Die Reviewer werden darüber informiert, dass sie ein Review zu schreiben haben.
		1.2	Es sind bereits Reviews zu der Frage vorhanden.	Die bereits vorhandenen Reviews werden auf unbearbeitet gesetzt. Die Reviewer werden darüber informiert.
2.0	Reviewer erstellt ein Review	2.1	Der Reviewer füllt alle Felder vollständig aus.	Das Review wird auf bearbeitet gesetzt und in der Datenbank gespeichert. Zudem wird vermerkt das wievielte Review der Reviewer gegeben hat.
		2.2	Der Reviewer vergisst, Felder auszufüllen.	Der Reviewer erhält eine Meldung, dass er vergessen hat manche Felder auszufüllen.
3.0	Reviewer betrachtet Reviews	3.1	Der Reviewer versucht Reviews einzusehen	Der Reviewer erhält eine Übersicht nur über das von ihm gegebene Review.
4.0	Autor betrachtet Reviews	4.1	Es sind Reviews vorhanden	Der Autor erhält eine Übersicht über alle bereits gegebenen Reviews, allerdings kann er die Informationen zu den Reviewern nicht einsehen.
5.0	Administrator betrachtet Reviews	5.1	Es sind Reviews vorhanden	Der Administrator erhält eine vollständige Übersicht über die gegebenen Reviews.

### 3.2.2 Fragepool



Nr.	Anwendungsfall	Nr.	Anfrage	Antwort
1.0	Autor löscht eine Frage	1.1	Reviews vorhanden	Zugehörige Reviews werden gelöscht

## 4 Lösungsstrategie und Entwurfsentscheidungen

### 4.1 Fachlicher Kontext

Für die Reviews sind zwei Plugins notwendig. Zum einen wurde ein Plugin entwickelt, das die Verwaltung von Reviews ermöglicht, zum anderen wurde ein Plugin für Reviewbare Multiple Choice Fragen erstellt. Es soll als Beispiel dienen, wie in ILIAS vorhandene Fragentypen erweitern werden können, damit sie reviewbar werden. Für die Plugins wurden verschiedene Plugin-Slots verwendet. Das Fragen-Plugin ist ein `TestQuestionPlugin` und das Review-Plugin ist ein `RepositoryPlugin`.

### 4.2 Reviewbare Frage-Plugin

Das Plugin für reviewbare Multiple Choice Fragen stellt eine Erweiterung der Multiple Choice Frage dar. Diese wird dabei um die Attribute der Taxonomie und Wissensdimension nach Prof. Wollersheim erweitert. Die Taxonomiestufe einer reviewbaren Frage beeinflusst die Nutzung von den in ILIAS bereits vorhandenen Taxonomien nicht. Das Plugin soll außerdem als Vorlage dafür dienen, wie man Plugins für andere Fragentypen entwickelt, um diese ebenfalls reviewbar zu machen.

#### 4.2.1 Ordnerstruktur

Die Ordnerstruktur richtet sich nach den ILIAS-Konventionen<sup>1</sup> für das 'Test Question Plugin'. Die Datei `plugin.php` definiert die ID des Plugins (`revmc`) und die kompatiblen ILIAS-Versionen (`4.4.xx`), außerdem hält sie die aktuelle Version des Review-Plugins fest. Im Ordner `'classes'` befinden sich die `'...Plugin'`- und `'...Feedback'`-Klassen, welche von der ILIAS-eigenen Multiple-Choice-Frage übernommen wurden. In den Unterordnern `'import'` und `'export'` befinden sich die PHP-Dateien, welche für die Auslagerung und Wiedereinbindung von Fragen zuständig sind. Die Klassen `'assReviewableMultipleChoice'` und `'assReviewableMultipleChoiceGUI'` werden in der Klassenübersicht genauer betrachtet.

Im Ordner `'lang'` befinden sich die entsprechenden Language Dateien für Deutsch und Englisch, in denen den Variablen die Ausdrücke zugeordnet werden.

---

<sup>1</sup>[http://www.ilias.de/docu/ilias.php?ref\\_id=42&obj\\_id=27029&cmd=layout&cmdClass=ilLMPresentationgui&cmdNode=ih&baseClass=ilLMPresentationGUI&obj\\_id=64260](http://www.ilias.de/docu/ilias.php?ref_id=42&obj_id=27029&cmd=layout&cmdClass=ilLMPresentationgui&cmdNode=ih&baseClass=ilLMPresentationGUI&obj_id=64260)

Der Ordner 'sql' beinhaltet die Datenbank-Verwaltungs-Datei dbupdate.php, in welcher neue Tabellen angelegt werden.

Im Ordner 'templates' befinden sich die Template Dateien, die den Content von der Multiple Choice Frage übernehmen.

#### 4.2.2 Klassenübersicht

Alle Klassen werden von den Klassen des bereits vorhandenen Fragentyps per Vererbung abgeleitet.

##### **assReviewableMultipleChoice**

Attribute:

Die Klasse besitzt zwei zusätzliche Attribute für Taxonomie (taxonomy) und Wissensdimension (knowledge\_dimension). Für diese besitzt sie außerdem get- und set-Funktionen, die nach den in ILIAS üblichen Regeln benannt sind.

getQuestionType():

Diese Funktion muss implementiert werden und gibt den Fragentyp als String zurück. Weiterhin wurden die Funktionen zum Laden und Speichern von Fragen aus der Datenbank erweitert, um die Taxonomie und Wissensdimension mit einzubinden.

loadReviewDataFromDb() / saveReviewDataToDb():

Diese neuen privaten Funktionen werden dazu benutzt die Taxonomie und Wissensdimension in die Datenbank zu speichern und aus dieser zu laden. Die Daten werden in der Datenbank qpl\_rev\_qst abgelegt (siehe Datenbankübersicht).

loadFromDb() / saveToDb():

Die Funktionen zum Laden von Fragen aus der Datenbank und zum Speichern von Fragen in die Datenbank wurden um Funktionen ergänzt, die zusätzlich die Taxonomie und Wissensdimension verwalten. Dazu wurden die von ass MultipleChoice geerbten Funktionen überschrieben und neben den neuen Funktionen per parent wieder eingebunden.

delete():

Die von assMultipleChoice geerbte Funktion muss überschrieben werden, weil man zusätzlich zum normalen Löschen auch die Taxonomie und Wissensdimension aus der Tabelle qpl\_rev\_qst löschen muss.

toJSON():

Da die Frage neue Attribute hat, muss die Darstellung in JSON um diese Attribute erweitert werden.

## **assReviewableMultipleChoiceGUI**

`writePostData()`:

Füllt die Aufklappmenüs Taxonomie('taxonomy') und Wissensdimension('knowledge\_dimension') gemäß POST mit Werten aus.

`writeReviewData()`:

Gibt den POST für die Taxonomie und Wissensdimension zurück.

`editQuestion()`:

Wurde lediglich um den Funktionsaufruf von `populateTaxonomyFormPart()` erweitert und sonst von der Oberklasse übernommen.

`populateTaxonomyFormPart()`:

Überprüft, ob das Review-Plugin vorhanden ist und erstellt Aufklappmenüs für die Taxonomie und Wissensdimension.

`getDefaultTaxonomy()/getDefaultKnowledgeDimension()`:

Geben ein Array zurück, ausgefüllt mit den Werten für Taxonomie und Wissensdimension, die sie aus der Datenbank des Review-Plugins auslesen.

`checkAddInput()`:

Überprüft die Aufklappmenüs Taxonomie und Wissensdimension auf Validität, also ob sie richtig ausgefüllt wurden und nicht der Defaultwert drinsteht.

## **4.3 Review-Plugin**

### **4.3.1 Ordnerstruktur**

Die Struktur des Review-Plugins richtet sich nach der Vorgabe für Repository Object Plugins, weil es gegen diese Plugin-Schnittstelle geschrieben wurde. Die Datei `plugin.php` definiert die ID des Plugins (`xrev`) und die kompatiblen ILIAS-Versionen (`4.4.xx`), außerdem hält sie die aktuelle Version des Review-Plugins fest. Im Unterordner `classes` befinden sich die Klassen, die ILIAS von einem Repository Object Plugin fordert. Diese müssen dazu jeweils eine plugin-typspezifische Oberklasse, die der Funktionalität der betreffenden Klasse entspricht, erweitern. Durch dieses Prinzip können wir sehr viele ILIAS-Komponenten wiederverwenden und erreichen automatisch eine Trennung von Anwendungslogik und Benutzeroberfläche.

### **4.3.2 Klassenübersicht**

Die Anwendungslogik steht in der Klasse `iObjReview`, die von `iObjectPlugin` erbt, und `iObjReviewGUI`, das von `iObjectPluginGUI` erbt, ist der GUI-Controller (auf beide

Klassen wird unten detailliert eingegangen). Die Klasse `ilObjReviewAccess`, die Zugriffe überprüft, erweitert die ihr zugehörige Oberklasse `ilObjectPluginAccess` und implementiert als neue Funktionalität im wesentlichen die Überprüfung, ob ein Nutzer tatsächlich berechtigt ist, auf ein von ihm angefordertes Objekt zuzugreifen (siehe dazu den Gliederungspunkt Sicherheit). Um Review-Objekte in der Gruppenübersicht anzeigen zu können, ist die Klasse `ilObjReviewListGUI` nötig, die von `ilObjectPluginListGUI` erbt. Die Grundklasse `ilReviewPlugin`, die von `ilRepositoryObjectPlugin` erbt, muss laut Spezifikation nur den Namen des Plugins zurück geben und wurde entsprechend gering gehalten. Der Unterordner `lang` enthält die Sprachdateien in der Form `ilias_[Sprachkürzel].lang`, in denen die Texte der Benutzeroberfläche ausgelagert sind, um den Anforderungen an die Internationalisierung des Plugins gerecht zu werden (siehe dazu den Gliederungspunkt Internationalisierung). Im Unterverzeichnis `sql` befindet sich die Datei `dbupdate.php`, ein Datenbank-Updateskript, das bei jeder neuen Version des Plugins von ILIAS ausgeführt wird. Es beinhaltet das Anlegen aller Tabellen des Plugins in der ILIAS-Datenbank sowie die Füllung derjenigen Tabellen, die Enumerationen darstellen, mit ihren Einträgen. Der Unterordner `Templates` enthält die Ordner `images` und `default`. In `images` sind die Icons des Plugins abgelegt, während `default` Templates für die eigenen GUI-Klassen des Plugins erhält. Jene befinden sich im Verzeichnis `classes/GUI` und werden unten detailliert erläutert.

## **ilObjReview**

Diese Klasse repräsentiert ein Review-Objekt, das zusätzlich zu den Attributen seiner Oberklasse noch über eine `group_id`, die ID der Gruppe in der das Review-Objekt angelegt wurde, verfügt. Das von `ilObjectPlugin` geerbte Attribut `obj_id` wird verwendet, um die Daten verschiedener Review-Objekte, die in einer ILIAS-Installation existieren können, in der Datenbank auseinanderhalten zu können. Die Klasse `ilObjReview.php` überschreibt die Methoden `doCreate`, `doRead`, `doUpdate`, `doDelete` und `doClone` ihrer Oberklasse, die zur Aktualisierung der Datenbank bei der Erstellung, beim Öffnen, beim Bearbeiten, beim Löschen bzw. beim Klonen eines Review-Objekts aufgerufen werden.

`doCreate()`:

Die Methode `doCreate` liest dazu die Gruppen-ID, die bei der Erstellung eines neuen Repository-Objekts in einer Gruppe von ILIAS als Parameter angehängt wird, aus.

`doRead()`:

Die Methode `doRead`, die jedes Mal, wenn ein Nutzer auf ein Review-Objekt zugreift, aufgerufen wird, ruft zusätzlich die private Methode `syncQuestionDB` auf, die die Fragedatenbank von ILIAS an die des Review-Objekts angleicht.

`doDelete()`:

Die Methode `doDelete` scheint von ILIAS nicht aufgerufen zu werden, weshalb auf eine Implementierung verzichtet wurde.

syncQuestionDB():

Die Funktionsweise von syncQuestionDB ist die Folgende: zunächst werden alle Fragen aus den Fragepools der Gruppe, in der das Review-Objekt angelegt wurde, und alle Fragen aus der Fragendatenbank des Review-Objektes geladen jeweils in einer Datenstruktur gespeichert. Fragen, die in beiden vorkommen, werden auf ihren Timestamp überprüft; hat die Frage aus dem Fragenpool einen neueren, ist sie seit dem letzten Aufruf des Review-Objektes bearbeitet worden. Dementsprechend wird ihr Zustand auf 0 gesetzt und von den dieser Frage zugeteilten Reviewern werden neue Reviews angefordert (Aufrufe von \$iLDB->update), außerdem werden die Reviewer darüber benachrichtigt (Aufruf von notifyReviewersAboutChange). Ist eine Frage nur in einem Fragepool enthalten, dann wurde sie neu erstellt und wird der Datenbank hinzugefügt (Aufruf von \$iLDB->insert) und die Administratoren werden darüber informiert (Aufruf von notifyAdminsAboutNewQuestion). Ist eine Frage nur in der Datenbank des Review-Objektes vorhanden, dann wurde sie im Fragepool gelöscht und wird nun automatisch mitsamt der zur ihr gehörenden Reviews aus der Datenbank des Review-Objektes entfernt (Aufruf von \$iLDB->manipulateF) und die vormaligen Reviewer werden informiert (notifyReviewersAboutDeletion).

Der Großteil der weiteren Methoden sind Datenbankabfragen, die Reviews, Fragen oder andere Datensätze abrufen und als assoziative Arrays zurückgeben. Diese Methoden sind deshalb notwendig, da das Plugin mehrere Anwendungsfälle abdeckt, für welche Daten unter unterschiedlichen Kriterien ausgewählt und formatiert werden müssen.

allocateReviews() / storeReviewById():

Von besonderer Bedeutung sind die Methoden allocateReviews und storeReviewById. Erstere legt, nachdem der Administrator einer Frage Reviewer zugeordnet hat, leere Review-Datensätze in der Datenbank an und letztere aktualisiert einen solchen Datensatz, nachdem ein Nutzer ein Review-Eingabeformular ausgefüllt hat, um die eingegebenen Daten.

finishQuestion():

Die Methode finishQuestion aktualisiert den Zustand von Fragen, die akzeptiert werden, sodass sie von den anderen Funktionen des Plugins nicht mehr berücksichtigt werden. verschiedene Ladefunktionen: Die Methoden taxonomy, knowledgeDimension, expertise, rating und evaluation laden die jeweilige Enumeration aus der Datenbank, damit sie später in Aufklappmenüs verwendet werden können.

notify-Funktionen:

Wann immer ein Fall eintritt, der eine Benachrichtigung erfordert, wird direkt die zur Nachricht gehörige notify-Methode dieser Klasse aufgerufen. Sie ermittelt die Empfänger der Nachricht und übergibt diese gemeinsam mit der eigentlichen Benachrichtigung der privaten performNotification-Methode, die die eigentliche Systemnachricht erstellt und abschickt. Dabei steht der eigentliche Text in den Sprachdateien (siehe oben), während

die Methoden nur Keys für diesen Text übergeben.

## **ilObjReviewGUI**

Die Controller-Klasse für die GUI erbt fast alle Funktionalitäten von ilObjectPluginGUI, darunter auch eine Referenz auf die Anwendungsklasse ilObjReview.

`performCommand()` / `setTabs()`:

Die von anderen, ILIAS-internen Controllern aufgerufenen Methoden `performCommand` und `setTabs` führen einen Befehl aus bzw. zeigen die Tabs, in die die Benutzeroberfläche unterteilt ist, an. Dabei überprüfen sie jeweils, ob der Nutzer die erforderlichen Rechte hat.

`showContent()`:

Durch `getAfterCreationCmd` und `getStandardCmd` bestimmt, wird bei Aufruf des Review-Objektes zunächst der Befehl `showContent` ausgeführt, der eine Übersicht über Fragen und Reviews eines Nutzers erzeugt. Diese sind jeweils in externe GUI-Klassen ausgelagert, die unten erläutert werden.

`inputReview()` / `editProperties()` / `allocateReviews()` / `finishQuestions()`:

Bei Klicks auf die Tabelleneinträge oder die Tabs ermittelt ILIAS anhand der gespeicherten Link-Variablen den nächsten auszuführenden Befehl, der dann das Eingabeformular für den Nutzer bereitstellt. Das können die vier Befehle `inputReview`, `editProperties`, `allocateReviews` oder `finishQuestions` sein, die zunächst die GUI für das Review-Eingabeformular, Bearbeitung der Eigenschaften des Review-Objekts, Zuordnung von Reviewern zu Fragen bzw. Akzeptieren von gereviewten Fragen erstellen. Dabei verwenden sie die eigene GUI-Klasse `ilReviewInputGUI` bzw. die privaten Methoden `initPropertiesForm`, `initReviewAllocForm` und `initQuestionFinishForm`, die ihrerseits wieder auf eigene GUI-Klassen zurückgreifen.

`initPropertiesForm()` / `initReviewAllocForm()` / `initQuestionFinishForm()`:

Die Funktionen setzen die Ziele der Absenden-Buttons auf die zugehörigen Befehle `saveReview`, `updateProperties`, `saveAllocateReviews` bzw. `saveFinishQuestions`, die die eingegebenen Daten überprüfen und bei Erfolg mithilfe der entsprechenden Methode von `ilObjReview` in der Datenbank speichern.

`showReviews()`:

Ein weiterer Befehl dieses GUI-Controllers ist `showReviews`, der alle Reviews, die zu einer Frage gehören, mithilfe von `ilReviewOutputGUI` ausgibt.



### 4.3.3 Besonderheiten der eigenen GUI-Klassen

Im Unterverzeichnis `classes/GUI` wurde eine Reihe eigener GUI-Klassen in Form von Tabellen und Eingabeelementen erstellt, da die ILIAS-Komponenten nicht in allen Fällen genügt haben. Dabei nutzen sie die Template-Engine von ILIAS, um ihre visuelle Darstellung zu beschreiben, indem sie die HTML-Templates aus dem Ordner `templates/default` und die Stylesheets aus dem Verzeichnis `templates/default/CSS` einbinden.

#### **ilAspectHeadGUI / ilAspectSelectGUI**

`ilAspectHeadGUI` und `ilAspectSelectInputGUI` ermöglichen es, mehrere Textfelder (`ilNonEditableValueGUI`) bzw. Aufklappmenüs (`ilSelectInputGUI`) nebeneinander auf einer Zeile darzustellen. Zu diesem Zweck erweitern sie `ilCustomInputGUI`, in der die Grundfunktionalitäten für eigene GUI-Klassen bereits implementiert sind.

#### **ilCheckMatrixRowGUI**

`ilCheckMatrixRowGUI` funktioniert ähnlich, aber für Checkboxes, die für die Reviewer-Zuordnung in einer Matrix angeordnet werden, wobei vertikal die Fragen und horizontal die Gruppenmitglieder angeordnet werden. Für jede Frage existiert also eine Instanz von `ilCheckMatrixRowGUI`, die für jedes Gruppenmitglied eine Checkbox innehat. Die Postvariablen für die einzelnen Checkboxes werden im Konstruktor dynamisch erzeugt und können mittels `getPostVars` vom GUI-Controller abgerufen werden.

#### **ilQuestionFinishTableGUI / ilQuestionTableGUI / ilReviewTableGUI**

`ilQuestionFinishTableGUI`, `ilQuestionTableGUI` und `ilReviewTableGUI` erzeugen jeweils eine tabellanartige Ansicht der zu beendenden Fragen, der Fragen eines Nutzers bzw. der Reviews eines Nutzers anhand der bei der Erstellung übergebenen Datensätze. Dazu erweitern sie die ILIAS-Klasse `ilTable2GUI`.

#### **ilQuestionOverviewGUI**

`ilQuestionOverviewGUI` erzeugt lediglich eine Ansicht einer Frage, mit der der Nutzer nicht interagieren kann. Deshalb verwendet sie ausschließlich ein Template und erweitert keine ILIAS-Komponente.

#### **ilReviewInputGUI**

`ilReviewInputGUI` erzeugt das Review-Eingabeformular. Sie erweitert dazu `ilPropertyFormGUI`, die Standard-Formularklasse von ILIAS und erzeugt sowohl GUI-Elemente, die von ILIAS stammen, als auch eigene. Dazu müssen ihr der betreffende Reviewdatensatz und die Taxonomie der gereviewten Frage übergeben werden, außerdem erwartet sie die vom Plugin definierten Enumerationen als Parameter.

`setReadOnly()`:

Nach Aufruf der Methode `setReadOnly` können die GUI-Elemente von `ilObjReviewGUI` deaktiviert werden, sodass diese Klasse für die Review-Ausgabe wiederverwendet werden kann.

`checkInput()`:

Die Methode `checkInput` der Oberklasse wurde überschrieben, um bei den Aufklappenmenüs sicherzustellen, dass ein anderer Wert als der default-Wert ausgewählt wurde.

### **ilReviewOutputGUI**

Die Review-Ausgabeklasse `ilReviewOutputGUI` erweitert ebenfalls `ilTable2GUI`, um Review-Eingabeformulare in einer Tabelle darzustellen. Diese werden durch Instanzen von `ilReviewInputGUI` gebildet, die aufgrund eines Aufrufs von `setReadOnly` nicht mehr bearbeitet werden können. Diese Klasse erwartet zur Konstruktion ein Array von Reviews und ansonsten die gleichen Daten wie `ilReviewInputGUI`. Zum Erstellen der Ausgabe kann sie dadurch einfach über die gegebenen Reviews iterieren und für jedes von ihnen mit den gegebenen Daten ein Eingabeformular erstellen.

## **4.4 Datenbankstruktur**

Folgende Datenbanken werden durch die Plugins erstellt:

Name	Beschreibung
<code>qpl_rev_qst</code>	beinhaltet die Taxonomie und die Wissensdimension aller Fragen
<code>rep_robj_xrev_revobj</code>	Standardtabelle für Review-Plugin-Objekte, speichert Gruppen-Id
<code>rep_robj_xrev_quest</code>	speichert Review-spezifische Frage-Informationen
<code>rep_robj_xrev_revi</code>	speichert alle Review-Daten
<code>rep_robj_xrev_taxon</code>	beinhaltet alle Taxonomie-Stufen
<code>rep_robj_xrev_knowd</code>	beinhaltet alle Knowledge-Dimensionen
<code>rep_robj_xrev_rate</code>	beinhaltet Fragen-Bewertungsmöglichkeiten
<code>rep_robj_xrev_eval</code>	beinhaltet Bewertungsmöglichkeiten für einzelne Aspekte der Fragen
<code>rep_robj_xrev_expert</code>	beinhaltet alle möglichen Expertisen für Reviewern

## 5 Bausteinsicht

## 5.1 Entwurfsklassendiagramm



Das Entwurfsklassendiagramm besteht aus zwei Teilen, dem 'Reviewable Question'-Teil (blau) und dem 'Review'-Teil (hellbraun). Daneben ist noch eine Vielzahl an ILIAS-Klassen (grün) eingetragen, die alle Methoden enthalten, die von unseren Plugins aufge-

rufen werden. Lediglich auf grundlegende GUI-Komponenten wie beispielsweise Texteingabefelder wurde aus Gründen der Übersichtlichkeit verzichtet. Zur näheren Betrachtung liegt es als PNG-Datei im Verzeichnis 'Entwurf/Entwurfsdiagramme' vor.

Die Plugins können getrennt voneinander betrachtet werden, da die Interaktionen zwischen ihnen allein aus statischen Methodenaufrufen und Datenbankabfragen bestehen, die in einem Entwurfsklassendiagramm nicht dargestellt werden.

## 5.2 Reviewable Question

Die Klasse 'assReviewableMultipleChoice' erbt von 'assMultipleChoice' und ergänzt deren Datenbank-Methoden 'loadFromDb()' und 'saveToDb()' um die Verwaltung der Taxonomie und Wissensdimension.

Die Klasse 'assReviewableMultipleChoiceGUI' erbt von 'assMultipleChoiceGUI' und erweitert deren Funktionen '\_construct()', 'writePostData()' und 'editQuestion()' um die Taxonomie- und Wissensdimension-Angaben. Die Funktion 'populateTaxonomyFormPart()' fügt die Aufklappmenüs für die Taxonomie und Wissensdimension hinzu und übernimmt durch den Aufruf der Funktionen 'getDefaultTaxonomy()' und 'getDefaultKnowledgeDimension()' die Werte, die auch im Review-Plugin genutzt werden<sup>1</sup>. Die Funktion 'checkAddInput()' prüft, ob die Aufklappmenüs ordnungsgemäß ausgefüllt wurden und nicht der Default-Wert drin steht.

Die Klasse 'assReviewableMultipleChoiceExport' erbt von der 'assMultipleChoiceExport' und besitzt die Funktion 'toXML()', die die gleichen Befehle ausführt, wie die 'toXML()'-Methode der Oberklasse, allerdings wurde sie um den Teil erweitert, der dafür sorgt, dass auch die Taxonomien und Wissensdimension exportiert werden und die Methode 'addReviewMetadata()', die die Taxonomien in XML umformt.

'assReviewableMultipleChoiceImport' erbt von der 'assMultipleChoiceImport' und besitzt die Funktion 'fromXML()', welche die gleichen Befehle ausführt, wie die 'fromXML()'-Methode der Oberklasse, allerdings wurde sie um den Teil erweitert, der die Taxonomie und Wissensdimension ausliest.

Die Klassen 'ilAssReviewableMultipleChoiceFeedback' und 'ilAssReviewableMultipleChoicePlugin' erben von den Oberklassen 'ilAssMultipleChoiceFeedback' und 'ilAssMultipleChoicePlugin' und übernehmen all deren Funktionen.

---

<sup>1</sup>Achtung: Für das Funktionieren dieser Funktion muss das Review-Plugin zwingend installiert sein!

## 5.3 Review

Die Klasse `ilReviewPlugin` erbt von `ilRepositoryPlugin` und implementiert die Methode `'getPluginName()'`, die den Namen des Review-Plugins liefert.

Die Anwendungsklasse `ilObjReview` erbt von `ilObjectPlugin` und erweitert sie um das Attribut `group_id` und eine Getter- und Setter-Funktion dafür. Sie implementiert die Methode `'iniType()'` und überschreibt die Methoden zur Verwaltung des Datenbankeintrags eines Review-Plugin-Objekts, die mit `'do'` beginnen und von ILIAS aufgerufen werden. Weiterhin implementiert sie Methoden, die mit `'load'` beginnen und Datensätze aus der ILIAS-Datenbank laden. Die Methoden `'saveReview()'`, `'allocateReviews()'` und `'finishQuestions()'` dienen dazu, Nutzereingaben in der Datenbank zu speichern.

Über die statischen Methoden `'taxonomy()'`, `'knowledgeDimension()'`, `'expertise()'`, `'rating()'` und `'evaluation()'` können andere Objekte die von uns definierten Enumerationen aus der Datenbank laden, ohne eine Instanz von `ilObjReview` zur Verfügung haben zu müssen.

Die `'notify'`-Methoden dieser Klasse dienen zur Vorbereitung von ILIAS-Systemnachrichten, die anschließend durch einen privaten Aufruf von `performNotification` abgeschickt werden.

Der GUI-Controller `ilObjReviewGUI` erbt von `ilObjectPluginGUI` und erweitert sie um einige Standard-Methoden, die ILIAS auf GUI-Controllern aufruft, dies sind `'getType()'`, `'performCommand()'`, `'getAfterCreationCmd()'`, `'getStandardCmd()'` und `'setTabs()'`. Mithilfe der von ihr implementierten Methoden `'inputReview()'`, `'allocateReviews()'`, `'finishQuestions()'` und `'editProperties()'` werden Eingabeformulare erzeugt, deren Input dann durch die Methoden `'saveReview()'`, `'saveAllocateReviews()'`, `'saveFinishQuestions()'` bzw. `'updateProperties()'` in der Datenbank von ILIAS gespeichert wird.

Die mit `'init'` beginnenden Methoden dieser Klasse erstellen die eigentlichen GUI-Komponenten und die Methode `'showReviews()'` erzeugt ein Ausgabeformular.

`ilReviewListGUI` und `ilObjReviewAccess` erben von `ilObjectPluginListGUI` bzw. `ilObjectPluginAccess`. Sie erweitern diese Klassen um einige pluginspezifische Methoden, die im ILIAS Development Guide unter Repository Object Plugins <sup>2</sup> zu finden sind.

Die Klasse `ilObjReviewAccess` implementiert zusätzlich noch die statische Methode `'checkAccessToObject()'`, die vom GUI-Controller aufgerufen wird, um zu prüfen, ob ein Nutzer berechtigt ist, sich ein bestimmtes Formular anzeigen zu lassen.

Die GUI-Klasse `ilQuestionOverviewGUI` dient lediglich dazu, eine von ILIAS gelieferte Fragenvorschau mit zusätzlichen Daten anzureichern, was im Konstruktor passiert. Über die Methode `'getHtml()'` kann diese dann als Komponente in ein Formular eingebunden werden.

---

<sup>2</sup>[http://www.ilias.de/docu/goto\\_docu\\_pg\\_29962\\_42.html](http://www.ilias.de/docu/goto_docu_pg_29962_42.html)

Die GUI-Klassen `ilQuestionFinishTableGUI`, `ilReviewTableGUI`, `ilQuestionTableGUI` und `ilReviewOutputGUI` erzeugen jeweils Datentabellen. Dazu erben sie von `ilTable2GUI`, der Grundkomponente von ILIAS-Tabellen und überschreiben zwei ihrer Methoden: im Konstruktor bestimmen sie das Format der Tabelle, während sie in `'fillRow()'` festlegen, wie einzelne Zeilen mit Daten gefüllt werden. Letztere ist `protected`, weil sie nur durch `ilTable2GUI` aufgerufen werden soll.

Da die Datensätze von `ilReviewOutputGUI` komplexe Review-Eingabeformulare sind, hat sie zusätzlich eine Methode `'setUpData()'`, in der die Formulare erzeugt werden.

Die GUI-Komponenten `ilAspectSelectInputGUI`, `ilAspectHeadGUI` und `ilCheckMatrixRowGUI` dienen dazu, mehrere Aufklappenmenüs (`ilSelectInputGUI`), Textfelder (`ilNonEditableValueGUI`) bzw. Checkboxes (`ilCheckboxInputGUI`) nebeneinander in einem Formular darzustellen. Dazu erweitern sie die ILIAS-Klasse `ilCustomInputGUI` und nutzen die ILIAS-Template-Engine um mithilfe vorgefertigter Template-Dateien GUI-Objekte zu erstellen. Der Kontext, in dem diese GUI-Klassen eingesetzt werden, ist dabei unterschiedlich, weshalb sich ihre Methoden stark unterscheiden.

`ilAspectHeadGUI` benötigt beim Aufruf nur die Beschriftungen der Textfelder in Form eines Arrays und implementiert die Methode `'setDisabled()'`, um als Komponente in der wie im Composite Pattern organisierten GUI nutzbar zu sein.

`ilAspectSelectInputGUI` besitzt die privaten Attribute `disabled` in dem gespeichert ist, ob der Nutzer mit dem Objekt interagieren können soll und `select.inputs`, ein Array, das zu allen Aufklappenmenüs, die zu diesem Objekt gehören, die Enumerationen und den ausgewählten Wert speichert.

Da Setter-Methoden für beide Attribute nach ihrem Aufruf ein erneutes Rendern dieser GUI-Komponente erfordern, wurde das Füllen des Templates vom Konstruktor in die private Methode `'fillTemplate()'` verlegt, sodass alle Methoden dieser Klasse darauf zugreifen können.

`ilCheckMatrixRowGUI` speichert die ID der zur Zeile gehörenden Frage im Attribut `question_id` und die `$_POST`-Variablen der einzelnen Checkboxes, die aus Frage- und Nutzer-ID bestehen im Attribut `postvars`. Über Getter-Methoden werden beide für die Auswertung der Eingabe im GUI-Controller zugänglich gemacht.

`ilReviewInputGUI` stellt das Eingabeformular für Reviews dar und erbt als solches von `ilPropertyFormGUI`, der Grundkomponente von ILIAS-Formularen. Die Erzeugung einzelner Sektionen des Formulars ist vom Konstruktor in die privaten Methoden, die mit `'populate'` beginnen, ausgelagert. Die Methode `'setReadOnly()'` entfernt die Schaltflächen und ruft auf allen Unterkomponenten `'setDisabled()'` auf, wodurch das Eingabeformular für die Ausgabe wiederverwendet werden kann. Die Methode `checkInput` der Superklasse wurde überschrieben, da innerhalb des Formulars genutzte Aufklappenmenüs, in denen der Default-Wert angewählt ist, nicht als ausgefüllt gelten sollen.

## 6 Laufzeitsicht

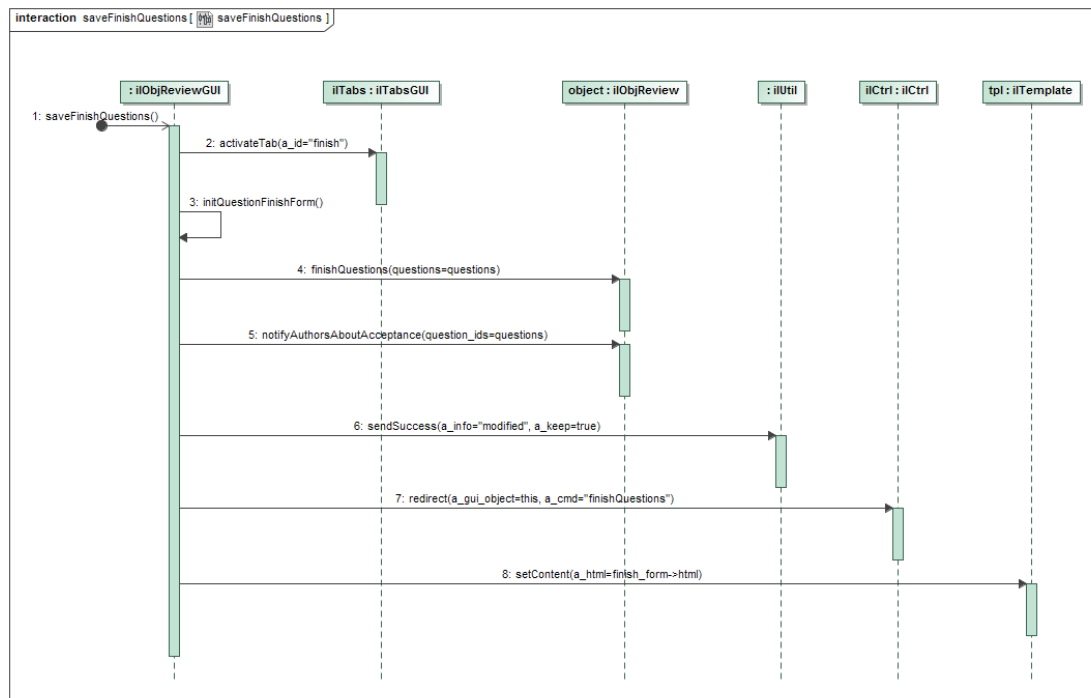
Die meisten Objekte, die während der Nutzung des Review- oder des Fragenplugins angelegt werden, werden direkt von ILIAS erzeugt. Dazu gehören vor allem die Hauptklassen, beim Reviewplugin im Ordner Review/classes abgelegt, zu denen auch die Anwendungslogik (ilObjReview) und der GUI-Controller (ilObjReviewGUI) zählen. ILIAS ruft dann auf diesen Objekten zahlreiche Methoden auf, wobei der wohl häufigste Fall die Methode performCommand des GUI-Controllers (ilObjReviewGUI) ist, die auf einen Befehl des Nutzers (z.B. Klick auf einen Button) folgt und ihrerseits wieder eine Methode von ilObjReviewGUI aufruft.

Weitere verwendete Objekte sind die globalen ILIAS-Objekte, die auch vom ILIAS-System erzeugt werden und von den Plugins referenziert werden können. Eine Erläuterung Wirkungsweise dieses Systems würde den Rahmen einer Dokumentation des Systems sprengen und ist für das Verständnis Laufzeitsicht auf die Plugins nicht nötig, weshalb an dieser Stelle auf den ILIAS Development Guide <sup>1</sup> verwiesen wird.

Am Beispiel der Methode saveFinishQuestions der Klasse ilObjReviewGUI, die immer dann aufgerufen wird, wenn ein Administrator den Abschluss der Reviewphase für eine Frage speichert, soll die Kommunikation der Plugin-Objekte untereinander und mit den globalen ILIAS-Objekten gezeigt werden.

---

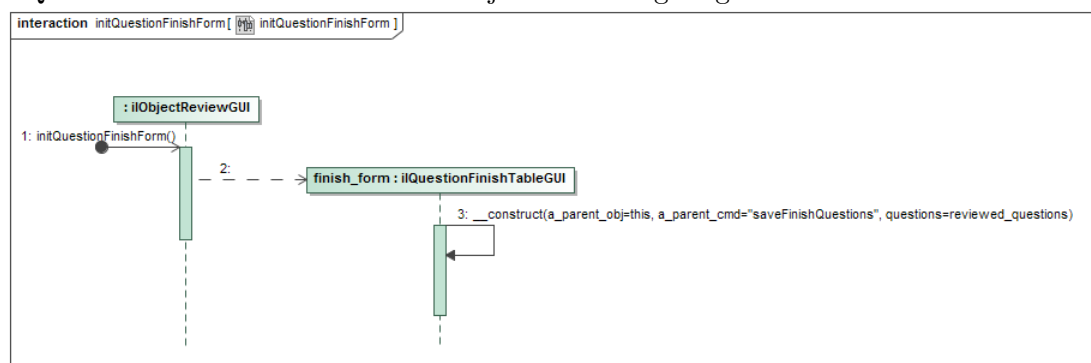
<sup>1</sup>[http://www.ilias.de/docu/goto\\_docu\\_pg\\_29964\\_42.html](http://www.ilias.de/docu/goto_docu_pg_29964_42.html)



Dabei werden aus Gründen der Übersichtlichkeit nur Methodenaufrufe, die direkt vom GUI-Controller `iObjReviewGUI` vorgenommen werden, dargestellt.

Die Instanz Anwendungs-klasse (`iObjReview`) steht dem GUI-Controller als Referenz im Attribut `object` zur Verfügung.

Auf die globalen ILIAS-Variablen wird durch deren Variablennamen zugegriffen, nachdem sie mithilfe des `'global'`-Statements von PHP als solche deklariert wurden. Diese Form der Kommunikation von Objekten ist auf alle anderen Methoden aller Klassen unserer Plugins übertragbar. Die einzige Ausnahme stellen die von uns selbst geschriebenen GUI-Komponenten dar, die vom Reviewplugin genutzt werden und sich im Ordner `Review/classes/GUI` befinden. Diese Klassen werden direkt vom GUI-Controller (`iObjReviewGUI`) instantiiert, wie im folgenden Sequenzdiagramm anhand der Methode `initQuestionFinishForm` der Klasse `iObjReviewGUI` gezeigt wird.





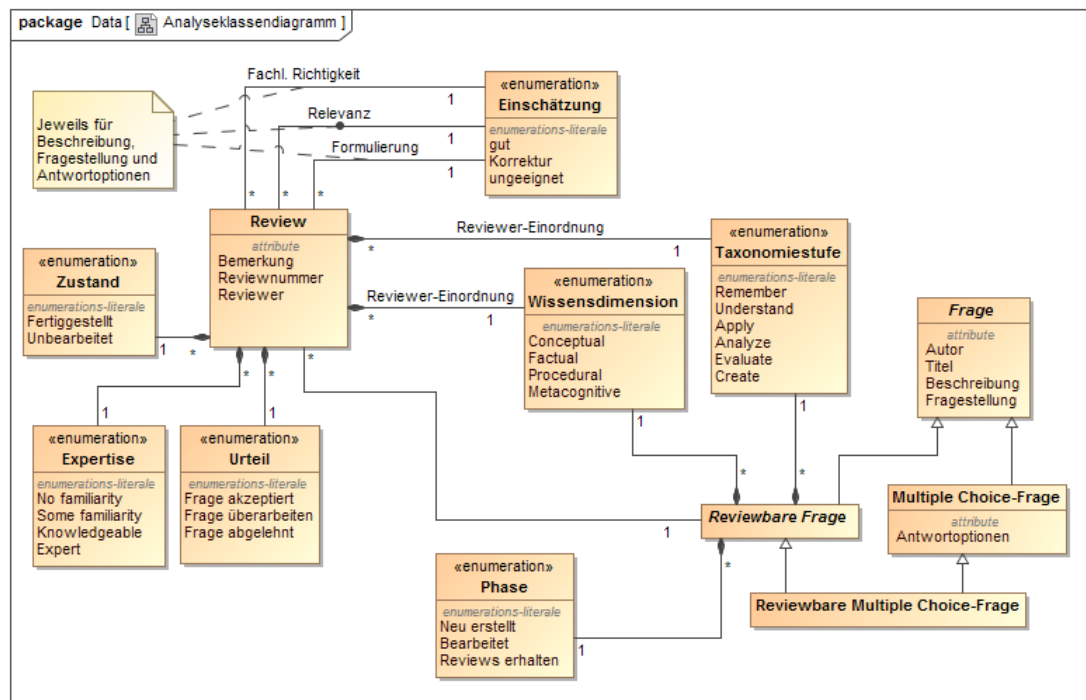
Diese Methode wird jedes Mal aufgerufen, wenn ein Administrator den Reviewzyklus für Fragen abschließen will und deshalb die zugehörige Ansicht aktiviert.

Um mit den anderen Objekten des Reviewplugins kommunizieren zu können, wird, falls nötig, eine Referenz auf die Instanz des GUI-Controllers als Parameter `a_parent_obj` an den Konstruktor der zu erstellenden GUI-Komponente übergeben. Die globalen ILIAS-Objekte werden auch bei unseren eigenen GUI-Komponenten mithilfe des `'global'`-Statements von PHP zugänglich gemacht.

# 7 Konzepte

## 7.1 Fachliche Strukturen und Modelle

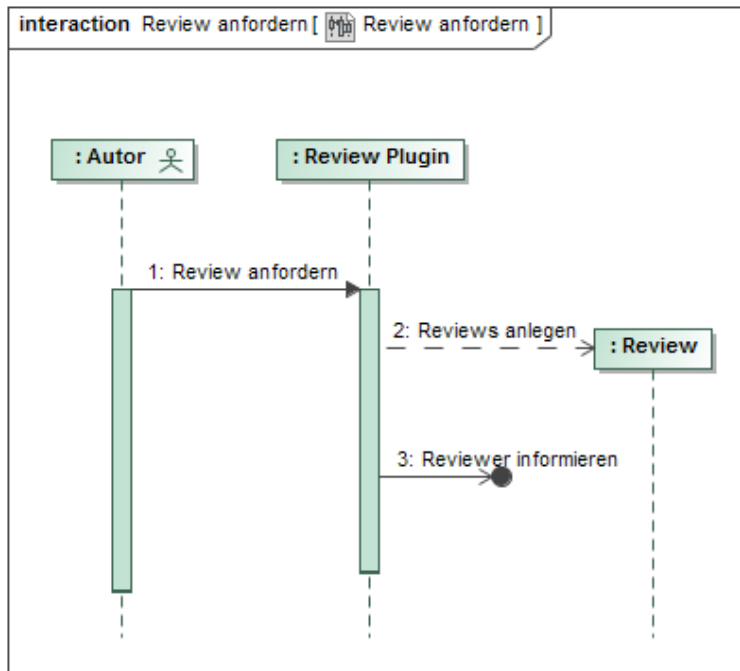
### 7.1.1 Analyseklassendiagramm



Der zentrale Bestandteil der Arbeit ist das Review. Es gehört zu einer reviewbaren Frage, wobei zu einer reviewbaren Frage mehrere Reviews erstellt werden können. Sowohl die reviewbare Frage als auch das Review haben eine Taxonomie und eine Wissensdimension. Das Review hat zusätzlich noch eine Einschätzung und ein Urteil, die letztendlich zur Bewertung dienen und einen Zustand, ob es bereits bearbeitet wurde oder nicht. Außerdem muss ein Reviewer, der ein Review erstellt, in jedem Review seine Expertise angeben, also wie gut er mit dem Thema der Frage vertraut ist. Eine reviewbare Frage hat entsprechend der Bewertungen eine Phase, in der sie sich befindet.

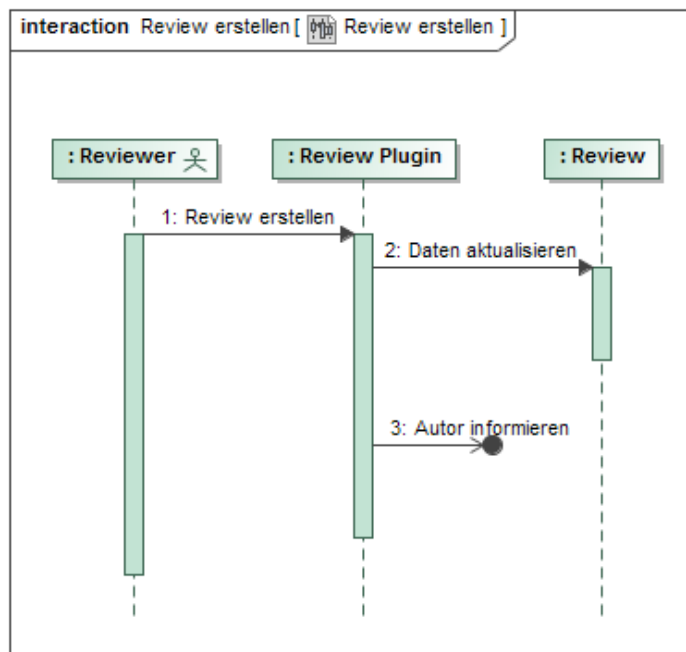
### 7.1.2 Sequenzdiagramm - Review anfordern

Nachdem der Autor eine Frage erstellt hat, kann er ein Review anfordern. Dadurch werden Reviewer bestimmt, welche vom Review-Plugin informiert werden, dass ein Review abzugeben ist.



### 7.1.3 Sequenzdiagramm - Review abschließen

Wenn ein Reviewer sein Review abgeschlossen hat, werden die Review-Daten formatiert und abgespeichert. Im Anschluss wird der Autor informiert, dass ein neues Review vorhanden ist.



## 7.2 Persistenz


Um Persistenz zu ermöglichen, nutzen wir die MySQL-Datenbankverwaltung, wie sie in ILIAS implementiert ist.

Alle Daten, die auf den einzelnen Seiten des Plugins dargestellt werden, laden mit mithilfe des Datenbank-Controllers von ILIAS, ilDB, als PHP-Arrays aus der Datenbank. Nutzereingaben werden über diese Schnittstelle ebenfalls direkt in die Datenbank geschrieben.

Datenbankanbindung im Development Guide: [http://www.ilias.de/docu/goto\\_docu\\_pg\\_25354\\_42.html](http://www.ilias.de/docu/goto_docu_pg_25354_42.html)

ilDB in der ILIAS-Dokumentation: [http://ildoc.hrz.uni-giessen.de/ildoc/Release\\_4\\_4\\_x\\_branch/html/d1/d20/classilDB.html](http://ildoc.hrz.uni-giessen.de/ildoc/Release_4_4_x_branch/html/d1/d20/classilDB.html)

## 7.3 Benutzeroberfläche

package Data[  InputForm ]

Frage

Beschreibung,  
Fragestellung?  
☐ Antwort 1  
☐ Antwort 2  
☐ Antwort 3

Review

	fachl. Richtigkeit	Relevanz	Formulierung
Beschreibung	<input type="text" value="gut"/>	<input type="text" value="gut"/>	<input type="text" value="gut"/>
Fragestellung	<input type="text" value="Korrektur"/>	<input type="text" value="Korrektur"/>	<input type="text" value="Korrektur"/>
Antwortoptionen	<input type="text" value="ungeeignet"/>	<input type="text" value="ungeeignet"/>	<input type="text" value="ungeeignet"/>

Anforderungsstufe und Wissensart

Einordnung durch Autor

Taxonomiestufe    Creating  
  
Wissensdimension    Procedural

Einordnung durch Reviewer

Taxonomiestufe      
  
Wissensdimension

Urteil

☐ Frage akzeptiert    ☐ Frage überarbeiten    ☐ Frage abgelehnt

Bemerkungen


Anmerkungen zur Überarbeitung // Begründung der Ablehnung

Reviewer

Name      
  
Expertise

Autor

Name    {sichtbar nur für Administrator}

package Data [  Plugin-Objekt ]

Meine Fragen

Neu erstellte Frage

Reviews einsehen

Frage mit angeforderten Reviews

Reviews einsehen

Bearbeitete Frage

Reviews einsehen

Meine Reviews

Unbearbeitetes Review 1

Review erstellen

Unbearbeitetes Review 2

Review erstellen

Fertiggestelltes Review 1

Review ansehen

package Data [  Reviewerzuordnung ]

Fragen	zuletzt geändert	Gruppenmitglieder	
Frage 1	01.11.14	Reviewerliste ▼	Frage akzeptieren
Frage 2	11.11.14	Reviewerliste ▼	Review anfordern
Frage 3	01.11.14	Reviewerliste ▼	Review anfordern
Frage 4	11.11.14	Reviewerliste ▼	Review anfordern

## 7.4 Ergonomie

Das Reviewplugin fügt sich nahtlos in die ILIAS Nutzererfahrung ein. Die Verwendung des Plugins ist verhältnismäßig intuitiv, die Funktionsweise durch eine Übersichtsseite schnell ersichtlich. Lediglich die einmalig notwendige Einrichtung erfordert einen gewissen Grad an technischen Verständnis, unter anderem des Gruppensystems von ILIAS. Die grafische Benutzeroberfläche des Plugins ist konsistent und darüberhinaus leicht verständlich sowie unanfällig für Anwenderfehler. Außerdem erhält der Benutzer Rückmeldung über von ihm getätigte Aktionen. Insgesamt ist die Funktionalität der Aufgabe gut angepasst und leicht handhabbar.

## 7.5 Transaktionsbehandlung

Datenbank-Transaktionen stehen in der von uns genutzten ILIAS-Version 4.4 nicht zur Verfügung und werden deshalb nicht von uns verwendet.

## 7.6 Sessionbehandlung

Die Nutzerinteraktion und Sessionbehandlung werden vom ILIAS-Kernsystem implementiert. User Data im Development Guide:

[http://www.ilias.de/docu/goto\\_docu\\_pg\\_204\\_42.html](http://www.ilias.de/docu/goto_docu_pg_204_42.html) Alle relevanten Daten des aktuellen Nutzers werden über eine Instanz der Klasse `ilObjUser` bereitgestellt. Diese Schnittstelle verwenden wir, um die ID des aktuellen Nutzers zu erfragen. `ilObjUser` in der ILIAS-Dokumentation: [http://ildoc.hrz.uni-giessen.de/ildoc/Release\\_4\\_4\\_x\\_branch/html/de/da1/classilObjUser.html](http://ildoc.hrz.uni-giessen.de/ildoc/Release_4_4_x_branch/html/de/da1/classilObjUser.html)

## 7.7 Sicherheit

Da das Plugin in das ILIAS-System eingebettet ist und die bereits erstellten ILIAS-Komponenten wiederverwendet, gelten dieselben Sicherheitsstandards. Nutzereingaben und Datenbankabfragen werden daher durch ILIAS-Komponenten überprüft und das Plugin passt sich nahtlos in die Benutzerverwaltung von ILIAS und ihr Rechtesystem ein. Eine mögliche Sicherheitslücke in ILIAS ist das Anhängen von IDs als GET-Parameter an die Adresse. Um diese zu schließen, wird vor jede Anzeige eines Formulars eine Validierung geschaltet, die überprüft, ob der Nutzer tatsächlich auf das Objekt, welches als GET-Parameter angehängt ist, zugreifen darf.

## 7.8 Plausibilisierung und Validierung

Wir greifen auf Nutzereingaben mittels der bereits implementierten GUI-Komponenten von ILIAS zu. Diese überprüfen die Nutzereingabe intern und somit transparent für unser Plugin. Die Benutzerauthentifizierung ist ebenfalls von ILIAS gedeckt, lediglich die unter 'Sicherheit' genannte Möglichkeit, dass der Benutzer bei Aufruf eines Formulars den ID-Parameter in der Adressleiste ändert und somit unberechtigt auf Daten zugreift, musste von uns ausgeschlossen werden.

## 7.9 Ausnahme-/Fehlerbehandlung

ILIAS übernimmt die Fehlerbehandlung.

Wenn ein Plugin auf das andere zugreift, prüfen wir, ob das andere Plugin aktiviert ist, um Zugriffsfehler zu vermeiden.

## 7.10 Logging, Protokollierung, Tracing

Wir haben diesbezüglich keine Maßnahmen getroffen, ILIAS bietet aber einen Developer-Mode, der Informationen über die aktuell aktiven Controller und eingebundenen Dateien bereitstellt bereitstellt.

## 7.11 Konfigurierbarkeit

Weder das Review-Plugin noch das Plugin für reviewbare Multiple Choice-Fragen sind konfigurierbar.

## 7.12 Internationalisierung

Das Plugin enthält ein englisches und ein deutsches Sprachpaket. Weitere Sprachen lassen sich über die ILIAS-Konforme Einbettung<sup>1</sup> hinzufügen, wozu die entsprechenden Sprachdateien in Review/lang bzw. ReviewableMultipleChoice/lang abgelegt werden müssen.

## 7.13 Testbarkeit

Das Reviewplugin ist in seiner Funktionsweise stark vom ILIAS System abhängig. Deswegen beschränkt sich die Testbarkeit auf das Testen der Korrektheit von Datenbankzugriffen. Die Klassen `assReviewableMultipleChoice` und `ilObjReview`, welche die Anwendungslogik des Reviewplugins bereitstellen, wurden ausführlich mittels Unit Tests getestet. Zu diesem Zweck kam das PHP Unit Framework `PHPUnit` zum Einsatz. Die Tests wurden methodenweise organisiert, so dass ein Testfall pro Methode existiert. Der Testvorgang setzt eine funktionierende ILIAS Installation voraus, da direkt auf die Datenbanktabellen des ILIAS Systems zugegriffen wird. Bestandteil jedes Testfalls ist die Konstruktion eines Datenbankszenarios, das durch die jeweilige Methode verändert und mit einem, entsprechend der zu erzielenden Funktionsweise der Methode vorgefertigten, Soll-Zustand verglichen wird.

## 7.14 Buildmanagement

Für das Plugin wurden PHP-Dateien geschrieben, die beim Starten von ILIAS in dessen Hauptdateien eingebunden werden. Es wurden 2 Repositorys genutzt, das `assReviewableMultipleChoice-Repository`<sup>2</sup> in welchem die Quellcode-Dateien für das Fragenplugin liegen und das `Review-Repository`<sup>3</sup>, welches die Dateien für das Review-Plugin enthält.

---

<sup>1</sup>[http://www.ilias.de/docu/ilias.php?ref\\_id=37&obj\\_id=8478&from\\_page=133&cmd=layout&cmdClass=illmpresentationgui&cmdNode=ih&baseClass=illMPresentationGUI&obj\\_id=128](http://www.ilias.de/docu/ilias.php?ref_id=37&obj_id=8478&from_page=133&cmd=layout&cmdClass=illmpresentationgui&cmdNode=ih&baseClass=illMPresentationGUI&obj_id=128)

<sup>2</sup><https://github.com/daelmo/assReviewableMultipleChoice>

<sup>3</sup><https://github.com/daelmo/Review>



## 8 Glossar

Diese Entwicklerdokumentation orientiert sich am arc42-Template, welches uns von Andy Püschel zur Verfügung gestellt wurde.