# Entwurfsbeschreibung

- Semantic Web: Plugin für Protégé -

GR-7, Version 0.5

Verantwortlicher: Sandro Wenzel

23.05.2005

## Inhaltsverzeichnis

# 1 Allgemeines

## 1.1 Kurzcharakterisierung

Das Protégé -Plugin **SCIL** (SoftConsultInstanceLister) ist ein dynamisches,interaktives und graphisches Plugin für das Ontologiewerkzeug Protégé , welches Instanzen einer Klasse in tabellarischer Form darstellt. **SCIL** ist für den Einzelnutzer– und Einzelplatzbetrieb konzipiert.

## 1.2 Systemvoraussetzungen

**SCIL** ist ein Plugin für Protégé und benötigt als solches eine Protégé Installation (Version 3.0). Zusätzlich wird das OWL Plugin (Version ???) für Protégé benötigt. Eine Java Installation (J2SDK) in der Version 1.4.2 wird benötigt.

## 1.3 Abgrenzung

- Das Plugin ist nicht netzwerkfähig.

# 2 Produktübersicht

## 2.1 Aufruf des Plugins

Zur Benutzung des Plugins **SCIL** muss zunächst Protégé gestartet und eine Ontologie geladen werden. Durch wechseln in das **SCIL** Tab wird das Plugin aktiviert.

## 2.2 Oberfläche

Die Oberfläche soll primär aus einer Tabelle (JTable) zur Darstellung der Instanzen bestehen. Über eine anklickbare Liste (JComboBox) und einen Selektionsbutton (JButton) kann man eine aktive Klasse (deren Instanzen dargestellt werden sollen) selekieren. Desweiteren gibt es eine Schaltfläche zum Laden/Aktualisierung der Instanzen in die Tabelle. Weiter Schaltflächen und Knöpfe sollen dazu dienen Sortier- und Filterkriterien festzulegen. All diese grafischen Komponenten werden auf einem JPanel platziert.

## 2.3 Funktionalität

Die grundlegende Funktionalität des Plugins wird anhand vom Zustandsdiagram in Abbildung 1 angedeutet. Die Abbildung orientiert sich dabei an den im Pflichtenheft beschriebenen Funktionen.

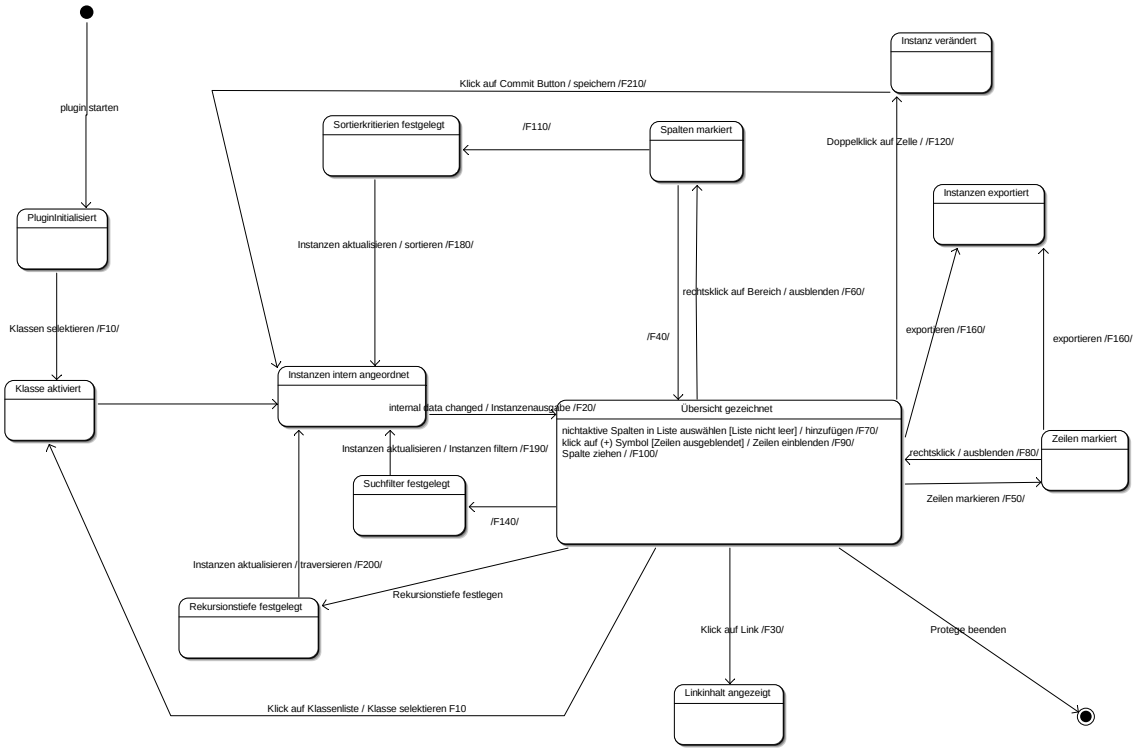


Abbildung 1: Zustandsdiagram/ Ablaufdiagram **SCIL** Plugin. Die gezeichneten Transitionen und Funktionalitäten entsprechen denen aus dem Pflichtenheft

# 3 Grundsätzliche Designentscheidungen

## 3.1 Allgemeines

Für die Modellierung des Plugins verwenden wir das **MVC** Designmuster welches eine Trennung der Applikation in **M**odel, **V**iew und **C**ontroller vornimmt. Dabei kommen den einzelnen Teilen folgende Aufgaben zu

- **Model:** beinhaltet Daten der Applikation sowie die grundlegenden Funktionen, um auf den Daten zu operieren.
- **View:** Darstellung der grafischen Oberfläche mit Daten des Modells. Die Aktualisierung der Daten wird über eine message (observer) registriert. Der View soll mit java swing Komponenten gestaltet werden.
- **Controller:** Abbildung der GUI Ereignisse und Benutzeranfragen auf die Methoden beziehungsweise Logik des Modells.

Die Interaktion dieser Teile wird in Abbildung 2 verdeutlicht.

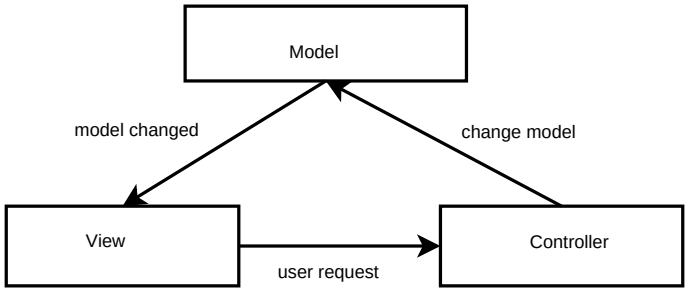


Abbildung 2: Das MVC Prinzip

## 3.2 Überblick Pakete und Klassen

### 3.2.1 Übersicht

Die Aufteilung der Klassen erfolgt zweckmässigerweise in die Pakete model, view und controller. Jedex dieser Pakete enthält eine Hauptklasse, die die anderen Komponenten des Paketes ansprechen und verwalten kann. Die Klassenamen werden entsprechend als **SCILModel**, **SCILController** und **SCILView** gewählt. Die Klasse **SCILPlugin** ist der eigentliche Container für das Plugin und als solcher von AbstractTabWidget abgeleitet. Es ergibt sich das Klassendiagramm in Abbildung 3.2.1.

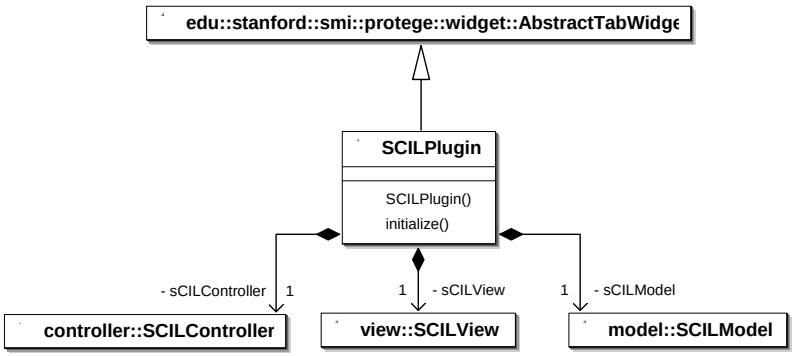


Abbildung 3: Klassenstruktur SCILPlugin

### 3.2.2 Verbale Beschreibung der Pakete

| Paket | Beschreibung | wichtige identifizierte Klassen |
|---|---|---|
| **view** | Das Paket **view** regelt die Darstellung des Plugins und die Anordnung aller grafischen Komponenten wie Buttons, Eingabefelder und vor allem die Instanzentabelle. Als graphischer Container wirkt die Klasse SCILView, die grundlegende Schaltknöpfe (Klasse selektieren, Instanzen darstellen, Commit Button), Auswahllisten (klassenSelektionsListe) und Textfelder (schlagwortSucheFeld) als Attribute hat. Sortierkriterien und Filterkriterien sollen in extra Bereichen (Panes) der GUI festgelegt werden. Diese verwalten dann ihre eigenen Komponenten wie FilterTextField undsoweiter. | • InstanzTabelle (JTable)<br>• SCILView<br>• SortierkriterienPane<br>• FilterkriterienPane |
| **controller** | Das Paket **controller** sorgt für die richtige Weiterleitung der Benutzeranfragen in Form von GUI-Ereignissen an das Modell. Dazu werden eine Reihe von Actions abgefragt und ActionListeners implementiert. | • SCILController<br>• KlasseSelectListener<br>• InstanzLadeListener<br>• FilterListener |

| Paket | Beschreibung | wichtige identifizierte Klassen |
|---|---|---|
| **model** | Das Paket **model** enthält sämtliche Klassen, die für die Datenhaltung und Operationen auf diesen Daten notwendig sind. Dies sind insbesondere Klassen der Ontologie und deren Instanzen und Properties. Instanzen können aktiv und passiv sein, je nachdem ob Sie im View dargestellt werden sollen order nicht. Zur Datenhaltung und für die Operationen dient ein Datencontainer (InstanzContainer) der alle Instanzen der aktiven Klasse enthält und sie verwaltet, sortiert, filtert. Für die Darstellung werden vom DatenContainer alle aktiven Instanzen an das InstanzTableModel übergeben, welches die Grundlage für die Tabelle im View ist. Dieses kennt auch die aktiven Properties, die in der Tabelle dargestellt werden sollen. Das Modell verwaltet weiterhin aktuelle Sortier- und Filterkriterien. Die Oberklasse SCILModel verwaltet alle anderen Klassen des Paketes und hat Zugriff auf die geladene Protégé Wissensbasis. Sie wird ausserdem ein Event generieren, falls sich Daten verändert haben. | • Istanz<br>• OntoKlasse<br>• Property<br>• InstanzContainer<br>• InstanzTableModel<br>• SCILModel<br>• ProtegeWissensbasis |

# 4 UML Modellierung

Ausgehend von der Partitionierung des Plugins in Pakete, ergibt sich folgende grobe Modellierung in Form von Klassendiagrammen.

## 4.1 Statische Aspekte

### 4.1.1 Paket Model

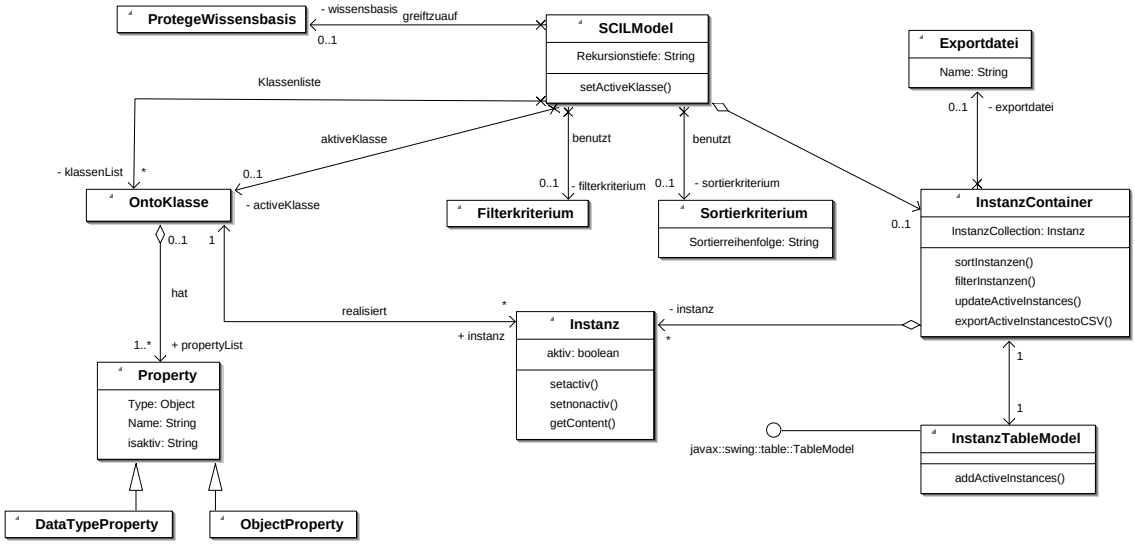


Abbildung 4: skizzenhaftes Klassendiagram des Paketes **model**

### 4.1.2 Paket View

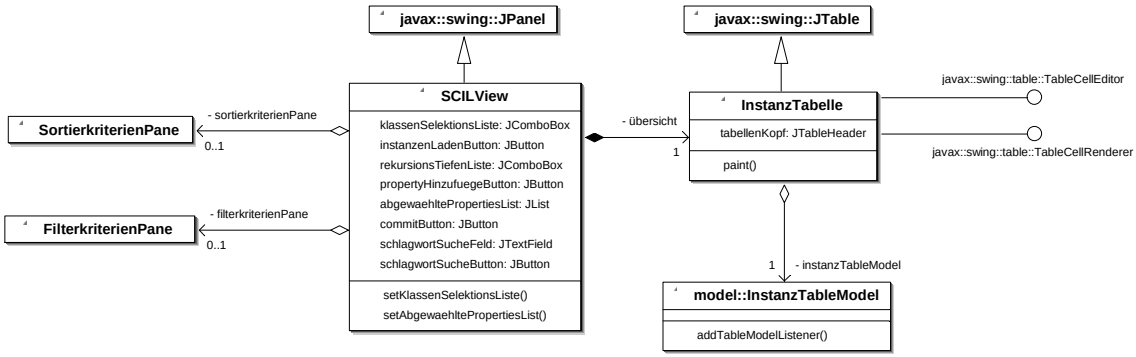


Abbildung 5: skizzenhaftes Klassendiagram des Paketes **view**

### 4.1.3 Paket Controller

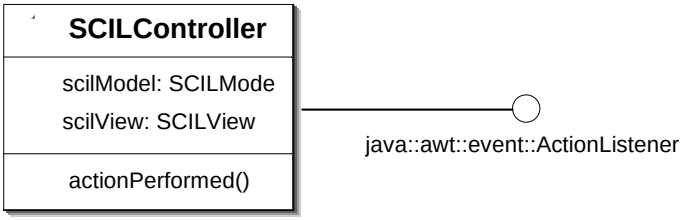


Abbildung 6: skizzenhaftes Klassendiagram des Paketes **controller**

## 4.2 Dynamische Aspekte

### 4.2.1 Kollaborationsdiagramm

Ein Beispiel für die grundlegende Interaktion zeigt das Diagram in Abbildung 7. Für andere Benutzeranfragen wie sortieren, filtern ergibt sich ein ähnliches Bild. Die Funktion updateActiveInstances() berücksichtigt jeweils die gesetzten Kriterien und ist so universell.

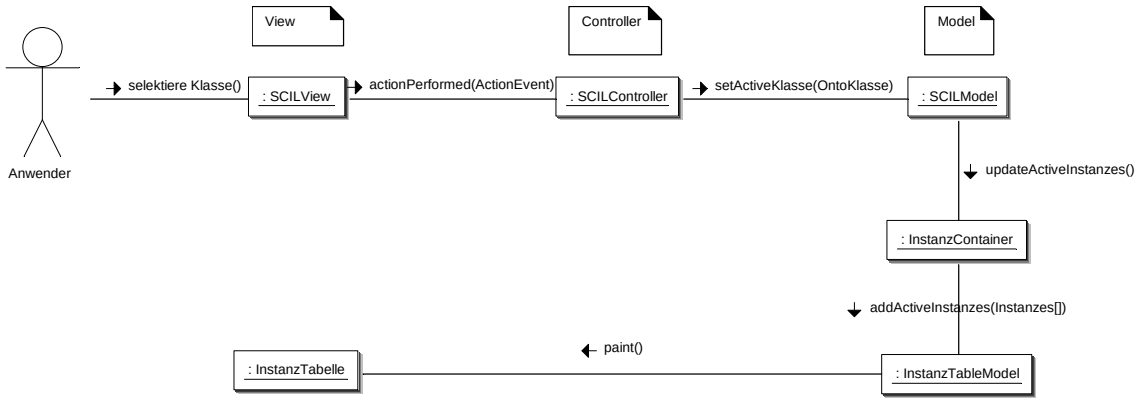


Abbildung 7: Beispielhaftes Kollaborationsdiagramm. Eine Klasse wird selektiert. Daraufhin werden über den Controller im Modell der InstanzenContainer aufgebaut/aktualisiert und das Tabellenmodell gebildet. Dieses sendet eine Nachricht zur Tabelle im View und die Instanzen werden dargestellt.