Technische Dokumentation für das PallBPMNRMELaunch-Plugin für Eclipse



FZI Forschungszentrum Informatik

Version 1.0

Inhaltsverzeichnis

[1 Übersicht 3](#_Toc37207864)

[1.1 Anforderungen 3](#_Toc37207865)

[1.2 Das Plugin Ausführen 3](#_Toc37207866)

[1.3 Vorstellung enthaltener Konzepte 4](#_Toc37207867)

[1.3.1 Eclipse Launch-Plugin Entwicklung 4](#_Toc37207868)

[1.3.2 UserInterface-Entwicklung 4](#_Toc37207869)

[1.3.3 Ausgaben 5](#_Toc37207870)

[2 Aufbau des Plugins 6](#_Toc37207871)

[2.1.1 Architektur und Klassen mit Funktionen 6](#_Toc37207872)

[2.1.2 Rollenmodell/Graph - Hierarchie Beispiel 7](#_Toc37207873)

[2.1.3 PalladioACRAnnotator 11](#_Toc37207874)

[2.1.4 DFD-Stub Generation 11](#_Toc37207875)

[2.1.5 UserInterface 12](#_Toc37207876)

[3 Verwendete Ressourcen 13](#_Toc37207877)

# Übersicht

In diesem Dokument wird ein Teil einer Erweiterung aus dem Projekt-Kontext von Mobility2Grid vorgestellt. Dabei handelt es sich um ein Launch-Plugin für Eclipse. Der Sinn dieses Plugins besteht darin, den Prozess der Rollenmodell-Extraktion aus Usagemodellen mit IntBIIS\_LP Erweiterung anzustoßen. Die extrahierten Daten sollen danach in geeigneter Form ausgegeben werden.

## Anforderungen

Folgende Anforderungen an das Plugin wurden zu Beginn des Projektes gestellt:

* **IntBIIS­\_LP muss korrekt nach Anleitung im äußeren Eclipse installiert sein** (inkl. Palladio und anderen Abhängigkeiten). Siehe Dokumentation von IntBIIS\_LP
* Der User muss im inneren Eclipse ein Projekt mit IntBIIS\_LP und Palladio Modelle auswählen können welche für die Analyse relevant sind

Zweck:

* Das Plugin soll die gegebenen Modelle einlesen, ein Rollenmodell und weitere Modelle extrahieren in Form von HTML ausgeben, die Daten eine Datenbank schreiben und einzelne benötigte Information in geeigneter Weise ausgeben.

## Installation

Um das Projekt zu installieren und im Anschluss das Plugin starten zu können müssen die folgenden Schritte durchgeführt werden.

1. Installation von Eclipse 2019-12 R <https://www.eclipse.org/downloads/packages/release/2019-12/r>
2. Danach muss über Help > Install new software die Datei unter Dokumentation/eclipseSetup.p2f ausgewählt und alle Pakete installiert werden. Dabei werden Palladio, BPMN und alle erforderlichen Abhängigkeiten installiert.
3. Danach muss das IntBIIS\_LP Repository über den git Projektimport importiert werden.
4. Danach kann auch PallBPMNRME über git importiert werden.
5. Nun sollten alle erforderlichen Abhängigkeiten installiert und im Projekt verfügbar sein

## Das Plugin Ausführen

Beachte den Punkt Anforderungen.

Das Plugin innerhalb der Entwicklungsumgebung verfügbar machen in der es entwickelt wird, funktioniert nicht. Jedoch wurde für solche Scenarios in Eclipse vorgesorgt. Es kann von der aktiven Entwicklungsumgebung aus eine RunTime-Umgebung gestartet werden. In dieser werden dann die Änderungen des Projektes angewandt. Mit dieser Technik kann nicht nur getestet werden, wie sich das bisherige Projekt verhält, es kann auch debuggt werden. Das bedeutet es können Breakpoints gesetzt werden, Variablen eingesehen werden, eben alles was beim Debugger von Eclipse so dazugehört.

Die RunTime-Umgebung ist eine weitere Instanz von Eclipse, welche die Anpassungen oder Änderungen aus dem gestarteten Projekt mit integriert hat. Das bedeutet auch, dass sie einen eigenen Workspace hat, in den Projekte importiert werden müssen. In unserem Fall sollten es EMF-Modellierungs-Projekte sein. Da es sich, wie schon erwähnt, lediglich um eine angepasste Instanz von Eclipse handelt, muss nicht weiter darauf eingegangen werden, wie diese Projekte importiert werden. Deswegen wird in der Klick-Anleitung auch davon ausgegangen, dass importierte Projekte vorhanden sind.

Die RunTime-Umgebung (innere Eclipse) kann ebenso einfach gestartet werden, wie man es vom Ausführen eines Projektes gewohnt ist. Zu aller erst öffnet man Eclipse, und wählt das Projekt aus dann wird wie folgt vorgegangen:

*Rechtsklick > Run As > Eclipse Application > Es erscheint eine Fehlermitteilung (probleme mit org.apache.xmlrpc) >* Nun startet die zweite Instanz von Eclipse *>*

Zum starten von PallBPMNRME muss folgendes im inneren Eclipse gemacht werden auf einem Projekt indem alle verfügbaren IntBIIS\_LP und Palladio Modelle liegen:

*Projekt auswählen das analysiert werden soll > Rechtsklick (Nur auf dem Projekt, nicht auf Unterordnern) > Run As > Run Configuration > Select PallBPMNRME\_LP launch*

Es öffnet sich die run configuration von PallBPMNRME\_LP launch. Alle Modelle müssen korrekt ausgewählt werden. Die Anzahl der IntBIIS\_LP Modelle muss mindestens der ausgewählten BPUsagemModel gleichen. Anschließend muss apply und Rund gedruckt werden.

Die **Analyse Ergebnisse** landen im Ordner „Data Output“ im Projektverzeichnis im inneren Eclipse und beinhalten die verschiedenen Extraktionsmodelle in Form von HTML.

Der Fehler beim Start hat bisher noch keinerlei Probleme verursacht. Aus diesem Grund wurde dem nicht weiter nachgegangen.  
Falls Interesse an dem Fehler besteht: <https://bugs.eclipse.org/bugs/show_bug.cgi?id=511117>

## Vorstellung enthaltener Konzepte

Um das Plugin in seinem jetzigen Zustand zu verstehen, sind Vorkenntnisse in der Eclipse-Plugin-Entwicklung, sowie rudimentäre Kenntnisse in der UserInterface-Entwicklung mit Java nötig.

### Eclipse Launch-Plugin Entwicklung

In Eclipse können diverse Erweiterungen hinzugefügt werden um die unterschiedlichsten Aufgaben zu bewältigen. Da verwundert es, dass die Erstellung eines Launch-Plugins so schlecht dokumentiert bzw. mit nachvollziehbaren und/oder erschöpfenden Anleitungen versehen ist. Aus diesem Grund können nur wenig Empfehlungen ausgesprochen werden um sich in die Thematik einzuarbeiten.

Tutorials:

<http://www.vogella.com/tutorials/EclipseLauncherFramework/article.html>  
<http://www.eclipse.org/articles/Article-Launch-Framework/launch.html>  
<https://www.eclipse.org/articles/Article-Your%20First%20Plug-in/YourFirstPlugin.html>

Anzumerken ist allerdings, dass der Eigentliche Aufwand gar nicht so groß ist, denn die Eclipse-Foundation hat vieles so vorbereitet, dass mit geringem Aufwand ein Ergebnis erzielt werden kann.

### UserInterface-Entwicklung

Um das User-Interface zu gestalten, steht einem von hause aus SWT zur Verfügung. SWT steht für Standart Widget Toolkit und ist auch als genau das zu sehen. Es sind diverse fertige Elemente wie DropDown-Menüs, Buttons, CheckBoxen etc. enthalten. Diese können mit Üblichen mitteln wie Layouts organisiert werden. Um so das gewünschte aussehen zu erzielen.

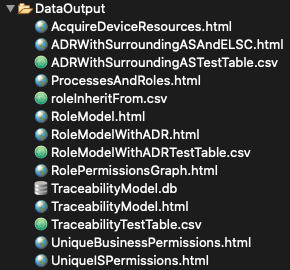
Der grobe Aufbau ist wie folgt:

Es gibt ein Träger-Fenster. In einem LunchPlugin ist dieses als >parent< bezeichnet. Auf diesen Träger können unterschiedliche Gruppen gelegt werden, welche dann mit den gewünschten Elementen beladen werden. An diese Elemente werden ActionListener gebunden. Diese Listener registrieren die Veränderung für die sie gedacht sind und lösen dann anschließend die hinterlegten Funktionen aus.

### Ausgaben

Während der Ausführung des Extraktionsprozesses werden in der Konsole alle benötigten Informationen zum Nachverfolgen des Extraktionsprozesses ausgegeben. Außerdem werden am Ende die Ergebnisse von eventuellen Tests bzw. Anderen nicht obligatorischen Analysen ausgegeben.

Im Ordner DataOutput werden nach jedem Durchlauf des Algorithmus die folgenden Dateien erzeugt:



* TraceabilityModel.db ist eine Kopie der während des Algorithmus erstellten Tabellen
* Die CSV Test Tabellen werden für Tests erzeugt und optional mit Referenzdateien verglichen
* Die HTML Dateien sind die verschiedenen Ausgaben der Extraktion wobei das TraceabilityModel alle extrahierten Daten vereint

# Aufbau des Plugins

In diesem Abschnitt liegt der Fokus auf den tatsächlich vorhandenen Elementen und welchen Zweck sie erfüllen. Die Struktur des Plugins besteht dabei aus den folgenden Packages:

extractor – enthält alle für die Ausführung und Extraktion relevanten Klassen und Methoden

visualizer – enthält alle für die Darstellung der extrahierten Daten relevanten Klassen und Elemente

### Architektur und Klassen mit Funktionen

Packages und Klassen des Plugins mit ihren Funktionen:

Alle Klassen und Packages mit ihren Funtkionen befinden sich in der JavaDoc unter /doc im Projektordner

**Package: db**

* **SQLiteJDBCDriverConnection.java:** 
  + Verwaltet JDBC Verbindung zur Datenbank
  + Die Datenbank befindet sich im Installationsverzeichnis von Eclipse
  + Stellt alle notwendigen Funktionen wie das Erstellen, Verbinden, Ausgeben und Befüllen der Datenbank zur Verfügung

**Package: extractor**

* **EMFModelReader.java**
  + Liest die per Pfad übergebenen Modelle im Eclipse Modeling Framework ein und gibt sie als Objekte zurück
* **PallBPMNRoleModelAnalyzer.java**
  + Liest aus den übergebenen Modellen alle erforderlichen Rollenmodelle und Objekte aus und schreibt diese in die Datenbank
* **Launch.java:** 
  + Enthält die Funktion, welche aufgerufen wird, nachdem der Run-Button der RunConfig geklickt wird.
  + Startet den Extraktionsprozess und alle anderen Analyse-und Ausgabeprozesse
  + Hier werden die Pfade aus der Konfiguration gelesen und durch alle BpusageModelle iteriert. Für jedes Modell wird dann der PallBPMNRoleModelAnalyzer aufgerufen und die entsprechenden Rollenmodelle extrahiert und in eine Datenbank geschrieben.
  + Am Ende wird der PallBPMNRMEVisualizer aufgerufen um die extrahierten Daten in HTML Tabellen zu schreiben
* **PalladioACRAnnotator.java**
  + Liest ein Modell ein und verändert die Werter des Modells programmatisch und speichert dies im Anschluss persistent im Modell ab

**Package: helper**

* **CSVUtils.java:** 
  + Helper Klasse um Javadaten in CSV Format umzuwandeln

**Package: model**

* **RoleModelPermissionList.java:** 
  + Enthält das Model für die Rollenmodelle samt Berechtigungen und Vererbungen

**Package: test**

* **CSVComparator.java:** 
  + Vergleicht zwei CSV Dateien und gibt die Unterschiede in der Konsole aus

**Package: ui**

* **LaunchConfigurationAttributes.java:**
  + Enthält die Attribute welche relevant sind um den Launch zu konfigurieren. Wenn diese Attribute nicht vorhanden sind, dann kann Eclipse nicht feststellen, ob sie verändert wurden. Wenn keine Veränderung festgestellt werden kann, dann wird der „Apply“-Button nicht Klickbar und somit können die Änderungen nicht genutzt werden. Es ist also unbedingt notwendig, dass hier die Attribute welche geändert werden können hier eigentragen und mit einem Default-Wert versehen werden.
* **SampleTab.java:**
  + In dieser Klasse wird das User-Interface erstellt. Außerdem werden die eingegebenen Werte zur weiteren Verarbeitung in die Konfiguration geschrieben.
  + Funktionen welche notwendig sind:
    - setDefaults(ILaunchConfigurationWorkingCopy): setzt die Default-Werte in LaunchConfigurationAttributes.java
    - initializeFrom(ILaunchConfiguration): überschreibt die Default-Werte in der LaunchConfiguration falls sie geändert werden
    - performApply(ILaunchConfigurationWorkingCopy): setzt die Werte aus der LaunchConfigurationAttributes fest, sodass Eclipse weiß, mit welchen gearbeitet werden soll.
* **SampleTabGroup.java:**
  + Setzt im Wesentlichen nur die Tabs/Reiter der LaunchConfig zusammen.
* **MultiCheckSelectionCombo.java**
  + Stellt eine MultiComboSelection Box mit allen nötigen Funktionen zur Verfügung

**Package: visualizer**

* **HTMLVisualizer.java**
  + Schreibt die übergebenen ArrayLists in HTML Tabellen und speichert diese im Dateisystem
* **PallBPMNRMEVisualizer.java**
  + Liest die verschiedenen benötigten Daten aus der Datenbank und speist in die verschiedenen Visualisierer ein.
* **GraphVisualizer.java**
  + Baut aus den Berechtigungen des Rollenmodells einen Graph zusammen und gibt diesen in einer HTML Datei aus
* **~~PDFVisualizer.java (Ausgaben unvollständig und werden nicht generiert)~~**
  + ~~Schreibt die übergebenen Daten in PDF Dateien und speichert diese im Dateisystem~~

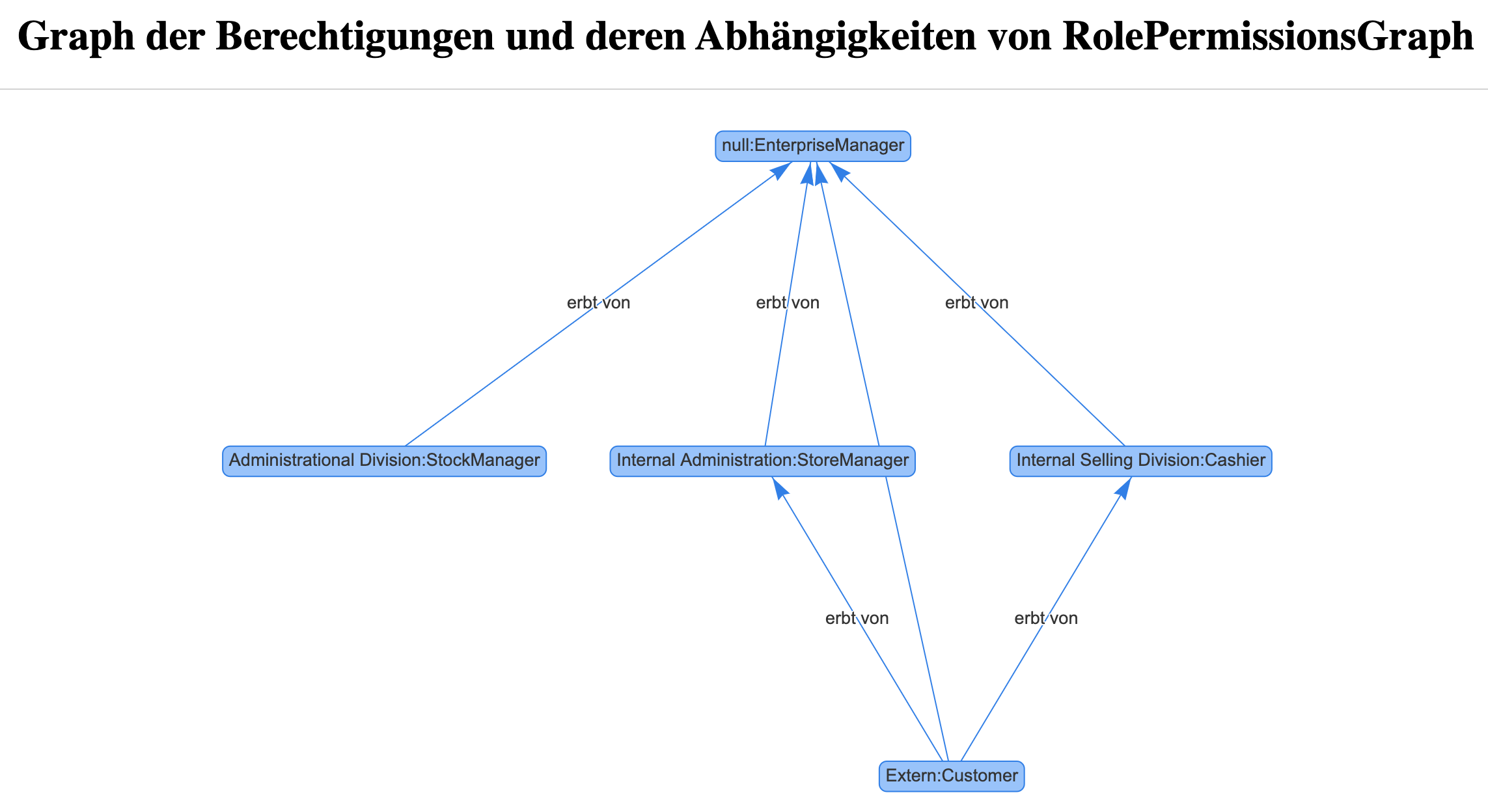
### Rollenmodell/Graph - Hierarchie Beispiel

Um eine grafische Übersicht der ausgelesenen Rollen und deren Vererbungen bzw. Hierarchien zu bekommen wurde eine neue Klasse GraphVisualizer.java implementiert die dies umsetzt. Dafür werden beim Launch, nachdem alle Modelle ausgelesen und ausgegeben wurden, die erforderlichen Berechtigungen samt Rollen ausgelesen und in einer neuen ArrayList gespeichert. Um die Datenstruktur der Rollen mit Berechtigungen und ihren Vererbungen sinnvoll darstellen zu können, wurde dafür die neue Modellklasse RoleModelPermissionList.java implementiert.

Danach wird die Klasse aufgerufen und in ihr eine HTML Datei unter CaseStudiesCoCoME/DataOutput/RolePermissionsGraph.html mit den entsprechenden Daten der RoleModelPermissionList erstellt.

Dabei wird die vis-network (<https://github.com/visjs/vis-network>) JavaScript Library verwendet.

Um ein aussagekräftiges Beispiel zu bekommen wurde der UseCase 10 so ummodelliert, dass eine sinnvolle beispielhafte Darstellung der einzelnen Abhängigkeiten möglich ist. Das Ergebnis des erzeugten Graphen sieht dann folgendermaßen aus:



Bei der Darstellung wurden in der verwendeten vis-network library die folgenden Optionen zur Darstellung verwendet:

var options = { layout: {

randomSeed: undefined, improvedLayout:true, clusterThreshold: 150,

hierarchical: { enabled:true, levelSeparation: 250,

nodeSpacing: 300, treeSpacing: 300, blockShifting: false,

edgeMinimization: true, parentCentralization: true,

direction: 'DU', // UD, DU, LR, RL

sortMethod: 'directed', // hubsize, directed

shakeTowards: 'leaves' // roots, leaves } } };

Unter [https://visjs.github.io/vis-network/docs/network/layout.html#](https://visjs.github.io/vis-network/docs/network/layout.html) lässt sich die Dokumentation der einzelnen Optionen nachlesen. Die folgende Tabelle zeigt alle verwendeten Optionen und ihre Dokumentation

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | Description |
| randomSeed | *Number* | *undefined* | When NOT using the hierarchical layout, the nodes are randomly positioned initially. This means that the settled result is different every time. If you provide a random seed manually, the layout will be the same every time. Ideally you try with an undefined seed, reload until you are happy with the layout and use the getSeed() method to ascertain the seed. |
| improvedLayout | *Boolean* | *true* | When enabled, the network will use the Kamada Kawai algorithm for initial layout. For networks larger than 100 nodes, clustering will be performed automatically to reduce the amount of nodes. This can greatly improve the stabilization times. If the network is very interconnected (no or few leaf nodes), this may not work and it will revert back to the old method. Performance will be improved in the future. |
| clusterThreshold | *Number* | *150* | Cluster threshold to which improvedLayout applies. |
| hierarchical | *Object or Boolean* | *Object* | When true, the layout engine positions the nodes in a hierarchical fashion using default settings. For customization you can supply an object. |
| hierarchical.enabled | *Boolean* | *false* | Toggle the usage of the hierarchical layout system. If this option is not defined, it is set to true if any of the properties in this object are defined. |
| hierarchical.levelSeparation | *Number* | *150* | The distance between the different levels. |
| hierarchical.nodeSpacing | *Number* | *100* | Minimum distance between nodes on the free axis. This is only for the initial layout. If you enable physics, the node distance there will be the effective node distance. |
| hierarchical.treeSpacing | *Number* | *200* | Distance between different trees (independent networks). This is only for the initial layout. If you enable physics, the repulsion model will denote the distance between the trees. |
| hierarchical.blockShifting | *Boolean* | *true* | Method for reducing whitespace. Can be used alone or together with edge minimization. Each node will check for whitespace and will shift it's branch along with it for as far as it can, respecting the nodeSpacing on any level. This is mainly for the initial layout. If you enable physics, the layout will be determined by the physics. This will greatly speed up the stabilization time though! |
| hierarchical.edgeMinimization | *Boolean* | *true* | Method for reducing whitespace. Can be used alone or together with block shifting. Enabling block shifting will usually speed up the layout process. Each node will try to move along its free axis to reduce the total length of it's edges. This is mainly for the initial layout. If you enable physics, the layout will be determined by the physics. This will greatly speed up the stabilization time though! |
| hierarchical.parentCentralization | *Boolean* | *true* | When true, the parents nodes will be centered again after the layout algorithm has been finished. |
| hierarchical.direction | *String* | *'UD'* | The direction of the hierarchical layout. The available options are: UD, DU, LR, RL. To simplify: up-down, down-up, left-right, right-left. |
| hierarchical.sortMethod | *String* | *'hubsize'* | The algorithm used to ascertain the levels of the nodes based on the data. The possible options are: hubsize, directed.  Hubsize takes the nodes with the most edges and puts them at the top. From that the rest of the hierarchy is evaluated.  Directed adheres to the to and from data of the edges. A --> B so B is a level lower than A. |
| hierarchical.shakeTowards | *String* | *'roots'* | Controls whether in directed layout should all the roots be lined up at the top and their child nodes as close to their roots as possible (roots) or all the leaves lined up at the bottom and their parents as close to their children as possible (leaves, default). |

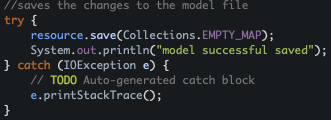
### PalladioACRAnnotator

Der PalladioACRAnnotator dient zum Auslesen und programmatischem Verändern von Modellen. Diese werden dann persistent im Dateisystem gespeichert. Im Fall des PalladioACRAnnotator werden .repository Modelle eingelesen und verändert. Dafür müssen im Userinterface (optional) die entsprechenden Modelle in folgendem Feld ausgewählt werden:

In den Modellen werden dann die Interfaces mit ihren jeweiligen Parametern und Returntypen ausgelesen und verändert. Als Beispiel werden momentan alle Operation signature Namen geändert und der jeweilige Return Wert auf einen Beispielwert aus den Datentypen gesetzt. Am Ende werden dann die veränderten Daten in die Ressource geschrieben:



Danach wir dann die Ressource mit dem folgenden Code (nach dem Tutorial: <http://www.vogella.com/tutorials/EclipseEMFPersistence/article.html#example>) persistent ins Dateisystem geschrieben:



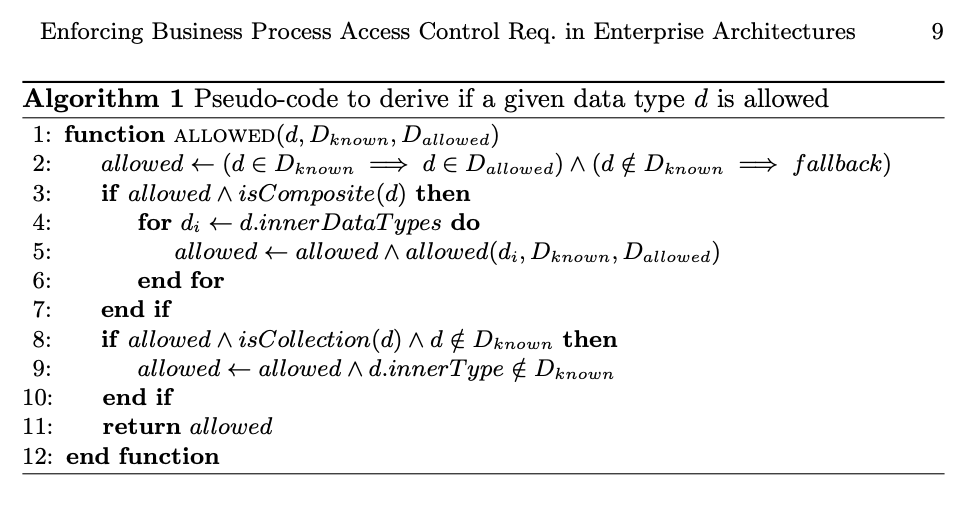
### DFD-Stub Generation

Beim Generieren der DFD Informationen muss in der UI des Plugins bei der Auswahl der Modelle ein Häkchen bei der Checkbox "Genreate DFD Information" gesetzt werden. Daraufhin ist die Auswahl von Repositorymodellen, Systemmodellen oder ACA nicht mehr möglich. Im Anschluss an die normale Rollenextraktion wird allerdings ein Backup der bestehenden Datenbank erstellt und in den Data Output Ordner uner /DataOutput/DFDInformation.db im Pluginprojekt kopiert. Danach werden noch die nicht relevanten Tabellen entfernt. Die kopierte Datenbank enthält dann nur noch die Tabellen:

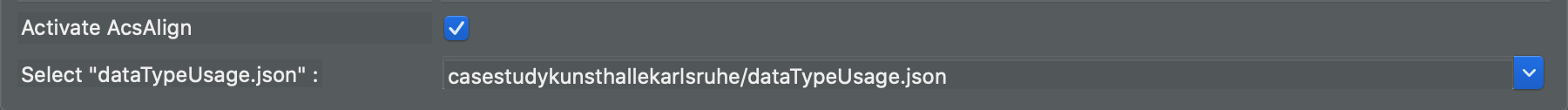
* ACRBP
* ACRAcquireRelease

### ACS Align Algorithmus

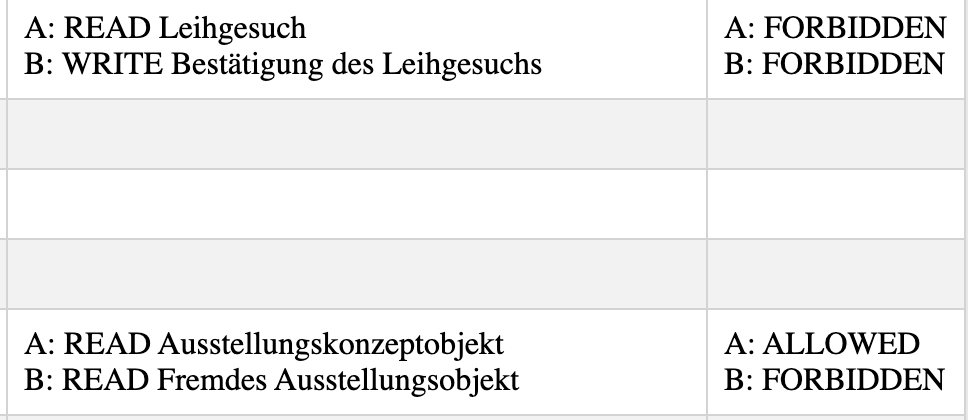
Beim Acs Align Algorithmus werden die READ/WRITE Datenflüsse aus einer gegebenen dataTypeUsage.json gelesen und im Anschluss mit den einzelnen IS Datentypen aus der zuvor erstellten ACRBP Tabelle verglichen. Der Algorithmus ist im folgenden Pseudocode einmal klar und übersichtlich dargestellt um besser nachvollzogen zu werden:



Das Ergebnis kann entweder Allowed oder Forbidden sein und wird als zusätzliche Spalten in einer neuen Tabelle gespeichert. Der Algorithmus wird ausgeführt, wenn in der Launch UI die Checkbox für Activate ACS Align aktiviert und eine gültige JSON Datei ausgewählt wurde.



Der Algorithmus geht so vor, dass er sich zuerst alle erlaubten Datentypen aus der ACRBP Tabelle holt (getAllAllowedDataTypes()), dann über diese iteriert und für die einzelnen Einträge dann durch die JSON Datei iteriert und sich die entsprechenden Einträge anhand der gemeinsamen ELSC Id herauszieht (getReadWriteDataTypesFromJson). Dann wird mit diesen Datentypen jeweils der Algorithmus aufgerufen (acsAlignAlgorithm) und eine entsprechende Permission als neue Spalten zu ACRBP hinzugefügt (inserACRBreachToTable()) und als neue Tabelle ACRBReach gespeichert. Das Ergebnis dieser Tabelle kann zum Beispiel wie folgt aussehen wobei alle Werte von ACRBP bestehen bleiben:



Zusätzlich wird eine ACSAlignLogFile Logdatei erzeugt in der alle Ergebnisse und Zwischenstände des Algorithmus geloggt und somit klar nachvollziehbar sind. In den nachfolgenden Zeilen ein Beispiel einer typischen Logausgabe:

INFORMATION: ------------------------------------

Sep. 09, 2020 3:48:53 NACHM. extractor.Launch acsAlignAlgorithm

INFORMATION: R/W DataType (STRING) Name: dAllowed= false; dKnown= false; allowed= true

Sep. 09, 2020 3:48:53 NACHM. extractor.Launch acsAlignAlgorithm

INFORMATION: R/W DataType (STRING) Name: Final allowed= ALLOWED

Sep. 09, 2020 3:48:53 NACHM. extractor.Launch acsAlignAlgorithm

INFORMATION: Analyze inner composite datatype

Sep. 09, 2020 3:48:53 NACHM. extractor.Launch acsAlignAlgorithm

INFORMATION: ------------------------------------

### UserInterface

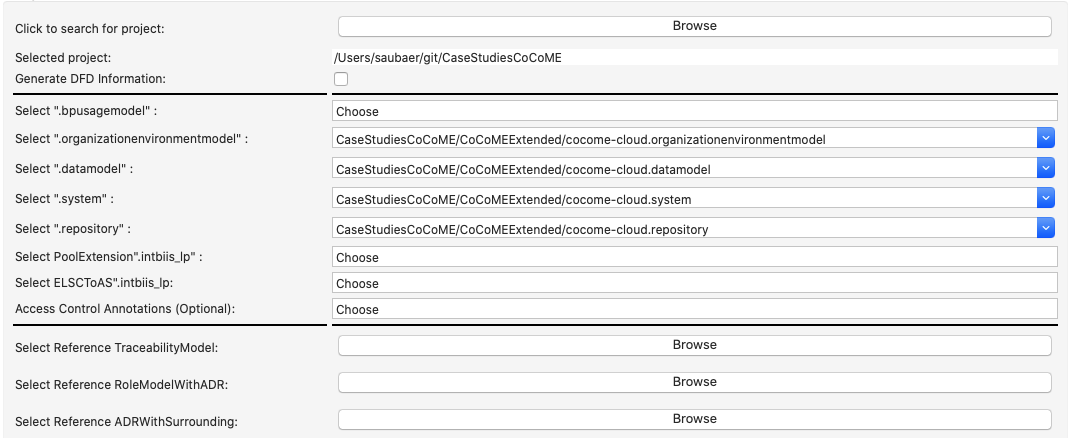
Das vorhandene Userinterface ist rudimentär gehalten und besteht aus nur einem relevanten Reiter mit einer Gruppe.

In dieser Gruppe ist eine Manuelle Auswahl der zu prozessierenden Projektes enthalten. Die Projektauswahl ermöglicht es, ein Projekt zu wählen, falls die RunConfig über das TaskMenü geöffnet wird.

Sollte die RunConfig jedoch über einen Rechts-Klick geöffnet werden, so wird das ausgewühlte Projekt erkannt und die DropDowns enthalten dann gleich schon die korrekten Dateien die sich in dem Projekt befinden. Die Dateien die in den DropDowns enthalten sind, Kategorisieren sich nach der Dateiendung. So kann keine falsche Datei ausgewählt werden, solange nicht mehrere Modelle innerhalb des Projektes zu finden sind. Andernfalls muss der Nutzer den Überblick behalten und die richtige Datei aus dem Pool der Dateien mit der gleichen Endung auswählen.

Wenn der Benutzer auf Browse klickt, kann er manuell einen anderen Projektordner auswählen, indem sich alle Modelldateien befinden müssen. Diese werden dann automatisch in den entsprechenden Feldern verfügbar sein und können ausgewählt werden.

Hier das finale Userinterface zum Auswählen der erforderlichen Informationen:



# Verwendete Ressourcen

|  |  |  |
| --- | --- | --- |
| Ressource | Version | Beschreibung/ Nutzen |
| Eclipse | 2019-12 R | Stark erweiterbare IDE |
| Java | JDK 1.11 | Java Bibliothek |