Refaktorisierung einer Architekturanalyse für Vertraulichkeit

Alina Valta

Institut für Informationssicherheit und Verlässlichkeit (KASTEL)
Betreuender Mitarbeiter: M.Sc. Frederik Reiche

1 Aufgabenstellung

Die Aufgabe dieses Praktikums ist eine Refaktorisierung der Projekte Confidentiality4CBSE[1] und PCM2Prolog[2]. Die Aufgabe besteht dabei aus zwei Teilen: Die Entfernung des Profil-Mechanismus aus dem Confidentiality4CBSE Projekt und dem verfeinern des Modells, sodass der Informationsfluss genauer modelliert werden kann. Außerdem soll es sich bei dem Praktikum um eine Refaktorisierung handeln, weshalb der PCM2Prolog Generator so angepasst werden muss, dass trotz verändertem Modell die Ausgabe als Prolog Prädikate unverändert bleiben.

2 Vertraulichkeitsanalyse

Die Idee hinter der Vertraulichkeitsanalyse für die Komponenten basierte Softwareentwicklung ist in einem technischen Bericht beschrieben worden [**kramer2017model**]. Es geht dabei darum eine Analyse über die Vertraulichkeit des Systems schon auf Architekturebene durchzuführen.

Dazu muss die Architektur im Palladio Komponenten Modell spezifiziert sein. Auf dieser Basis können dann mit Hilfe des Confidentiality4CBSE Projekts Vertraulichkeitsinformationen ergänzt werden. Durch eine Profil-Mechanismus können bestehende Palladio Komponenten mit Stereotypen annotiert werden, die Informationen über die Vertraulichkeit von Informationsflüssen an Interfaces oder von Operationen angeben oder Eigenschaften von Resourcen spezifizieren. Dabei kann es sich zum Beispiel um Maßnahmen zum Schutz von Server vor Angreifern handeln oder um Information welche Daten unverschlüsselt

[1] https://github.com/KASTEL-SCBS/Confidentiality4CBSE.git
[2] https://github.com/KASTEL-SCBS/PCM2Prolog.git

übertragen werden. Diese Profil verwendet Informationen aus dem Confidentiality-Modell indem Schutzmechanismen, Angreifer und Datenset definiert werden können die für die Architektur relevant sind.

Die Analyse erfogt dann in zwei Schritten: Zuerst werden die modellierten Vertraulichkeitsinformtionen in Prolog Prädikate übersetzt dies wird von dem Projekt PCM2Prolog umgesetzt. Danach werden die Prolog Prädikate untersucht und wenn es eine Vertraulichkeitsprobleme gibt werden diese textuell ausgeben. In diesem Praktikum ist sind nur die Schritte bis zu den Prolog Prädikaten relevant.

3 Modell

Das Modell besteht aus fünf Paketen: Data, Repository, System, Resources, Adversary. Diese Aufteilung wurde in der Refaktorisierung bei behalten. Das Profil bestand aus zwei Teilen, deren Funktion in dem neuen Modell den Paketen Repository und Resources hinzugefügt werden. Um den Profilmechanismus des Modelle zu entfernen mussten neue Entitäten eingeführt werden, welche wiederrum Referenzen auf Palladio Komponenten beinhalten.

3.1 Data

Das Data Package wird dazu verwendet um Informationen in DataSets zusammen zufassen. Modellierte Datenflüsse können dann diesen DataSets informationen hinzufügen. Außerdem gib es die Möglichkeit eine Gruppe von DataSets zu definieren: Eine DatasetMap. Diese kann zu einen dafür genutzt werden um ein DataSet für verschiedene Zuriffsrechte zu defninieren. Hierfür wird für jedes Zugriffsrecht ein SpecificationParameter definiert und für eine DataSetMap die verschiedenen Datensätze (ParamiziedDataSetMapEntry) mit zugewiesenem Zugriffsrecht. Zum anderen kann ein Datensatz auch für verschiedene User definiert werden, wenn modelliert werden soll, dass mehrere User das System verwenden und diese nur teilweise die Daten des anderen kennen dürfen. Dazu kann für eine Gruppe an Daten und einen User ein DataSetMapEnty erstellt werden.
Im ursprünglichen Modell hattet ein DataSetMapEntry kein eigenen Parameter um den User zu idenifizieren, sondern der Name des DataSetMapEntry wurde als Parameter angegeben. Um diese Referenzierung explizit zu machen wurde im neuen Modell die Klasse UserIdentifier hinzugefügt 3.1.

3.2 System

Im Package System wurde in der Klasse DataSetMapParameter2KeyAssignment das Attribute assigendKey von einem String zu einer Referenz zu UserIdentifier geändert 3.2 Im bisherigen Modell wurde hier der Name eines DataSetMapEntry eingetragen. Durch die Änderung wurde die Beziehung zwischen dem DataSetMapEntry und der DataSetMapParameter2KeyAssignment explizit modelliert. Dies Klasse modelliert, dass ein Connector ein User authentifiziert und ein bestimmten SpecificationParameter zu weißt. Die Stereotype

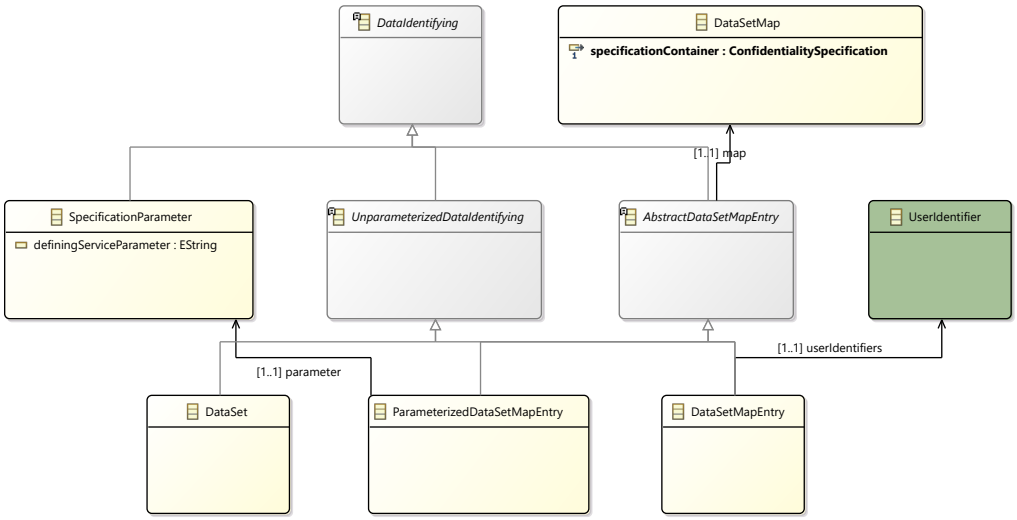

Abbildung 1: Genauere Modellierung des DataSetMapEntry

InformationFlowParameterEquation und InformationFlowParameterAssignment wurden entfernt, indem den Klassen SpecificationParameterEquation und AbstractSpecifiationParameterAssignment jeweils eine Referenz zu einem oder mehreren AssemblyContexten bzw. Connectoren hinzugefügt wurde.

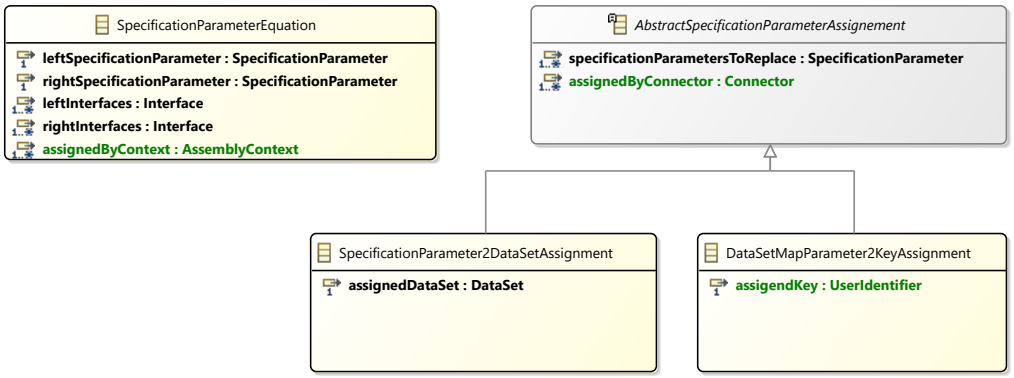

Abbildung 2: Referenzierung des UserIdentifiers

3.3 Repository

Der InformationFlow Stereotyp weist Interfaces oder Signatures ParameterAndDataPairs zu. Diese wiederum weisen ein oder mehrere DatenSets bestimmte Informationen dieser Schnittstellen oder Methoden zu, indem der String parameterSource entweder der Name eines Parameter oder \call, \return, * (alle Information) oder sizeof(<parameterSourceString>) ist. Diese Modellierung ist ungenau insbesondere, wenn mehrere Parameter den selben Namen haben, deshalb wurde im neuen Modell ein weiteres Paket Information 3.3 eingeführt, das die ParameterAndDataPair Klasse ersetzen soll. Die abstrakte Klasse Information

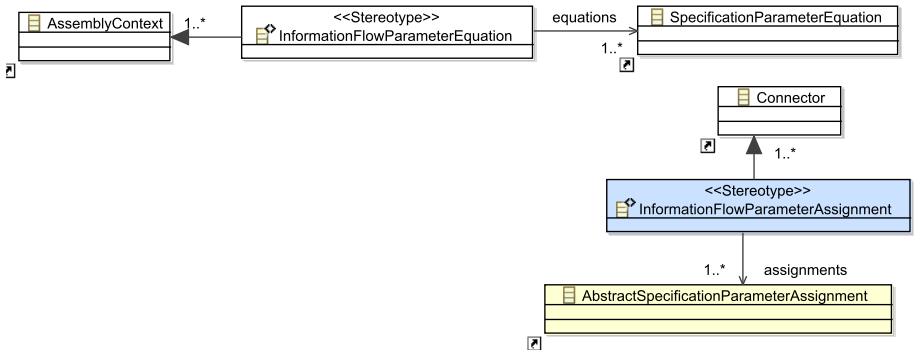

Abbildung 3: Referenzierung des UserIdentifiers

repräsentiert hierbei ein ParameterAndPair welche jedoch nur noch eine parameterSource haben kann. Diese Einschränkung wurde vorgenommen, da so die Unterklassen direkt den ParameterSource Wert ersetzen können z.B. ersetzt die Klasse CallInfomation ein ParamterAndDataPair mit der ParameterSource \call. Die Klasse PararameterInformation hat jetzt eine explizite Referenz auf einen Parameter. Diese Variante wurde gewählt, da in keinem der Beispiele mehr als eine ParameterSource für ein ParameterAndDataPair definiert wurde sondern immer mehre ParameterAndDataPairs erstellt wurden die teilweis die gleichen DataTargets haben und auf die gleiche Signatur angewandt wurden.
Der InformationFlow Stereotyp wurde entfernt indem eine abstrakte Klasse AbstractInformationFlow eingeführt wurde die mehrere Information Instanzen referenzieren kann und je nach Unterklasse auf ein Interface oder eine Signatur verweist. Im alten Modell konnte ein InformationFlow auch auf mehre Signaturen angewandt werden, dies ist im neuen Modell nicht vorgesehen, da die Information wie z.B. CallInformation für jede Signatur andere Informationen sind und deshalb als eigene Instanz der Klasse modelliert werden sollte und Referenzen zu Parameter nur sinnvoll sind wenn der InformationFlow auch nur auf die Signatur oder Interface dieses Parameters angewendet wird.
Die Stereotypen ServiceParameterAddition und InformationFlowParameter werden entfernt, indem die neue Klasse InformationFlowParameter einem AbstractInformationFlow einen AddedServiceParameter und mehrere SpecifiactionParamter zu weisen kann.

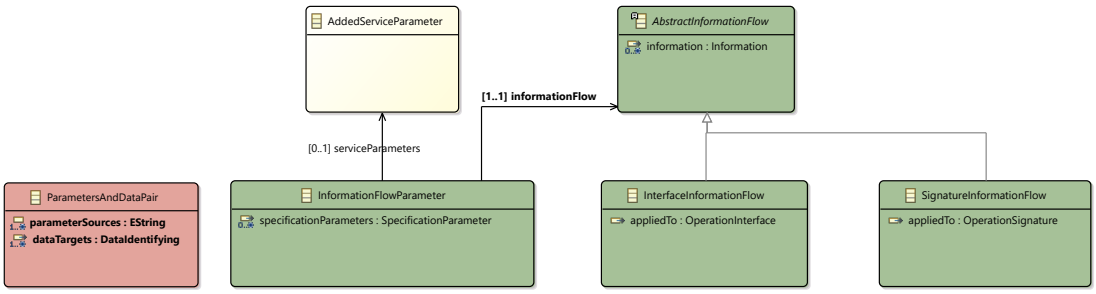

Abbildung 4: Referenzierung des UserIdentifiers

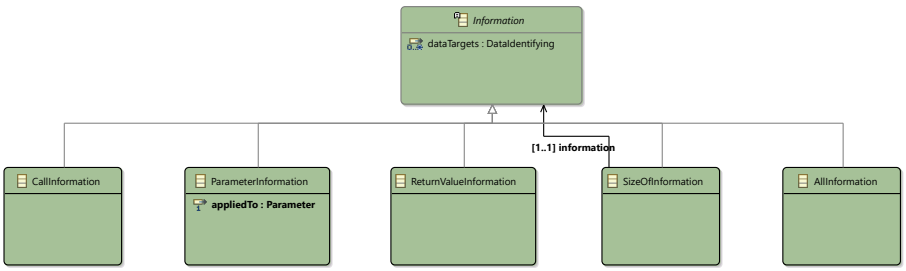

Abbildung 5: Referenzierung des UserIdentifiers

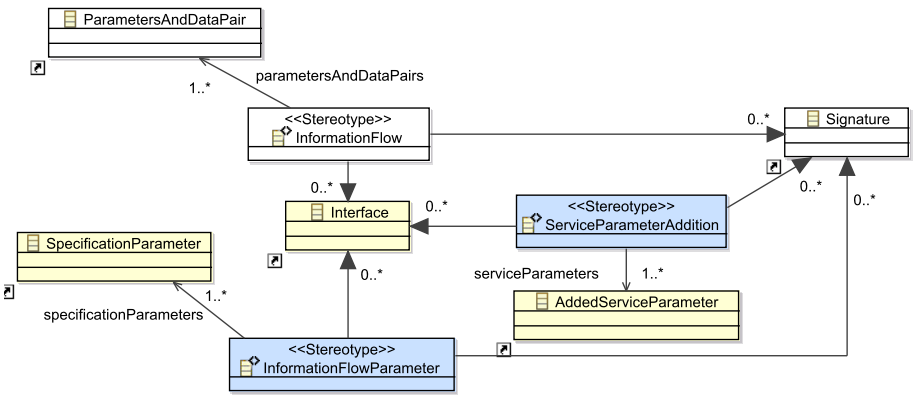

Abbildung 6: Referenzierung des UserIdentifiers

3.4 Resources

Die Stereotypen Sharing und FutherPhysicalConnections 3.4 wurde durch die Klasse ResourceContainerConfidentiality ersetzt. Der Stereotyp Encryption 3.4 wird ersetzt durch die Klasse Encryption. Die Klasse LocationsAndTemperProtectionsPair wird zur abstrakten Klasse AbstraktResourceProtection, deren Unterklassen LinkingResourceProtection und ResourceContainerProtection referenzieren auf LinkingResources bzw. ResourceContainer, sodass der Stereotyp LocationAndTamperProtection nicht mehr benötigt wird. Allerdsings kann eine AbstraktResourceProtection Instanz jetzt nur noch entweder auf LinkingResoucres oder auf ResourceContainer verweisen nicht mehr auf beides gleichzeitig.

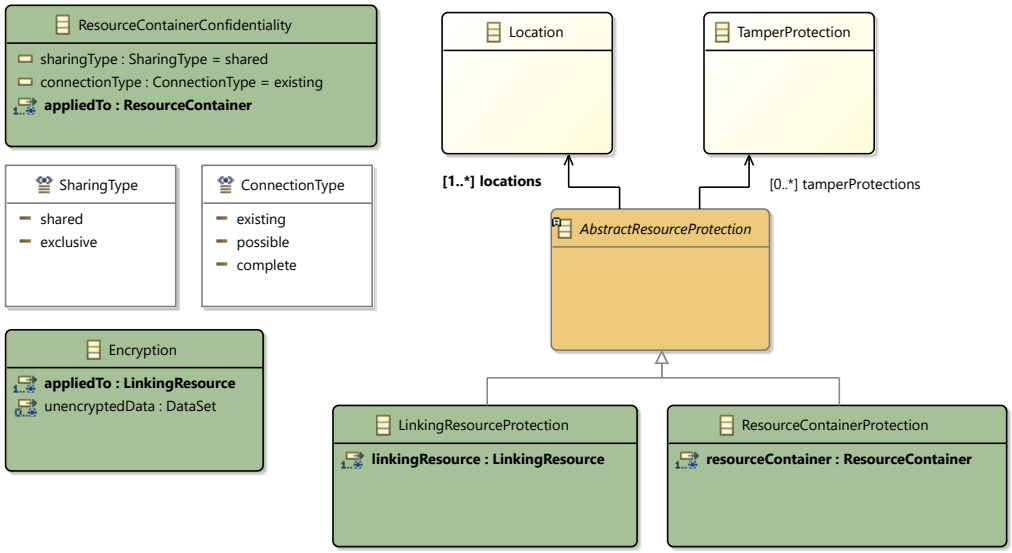

Abbildung 7: Referenzierung des UserIdentifiers

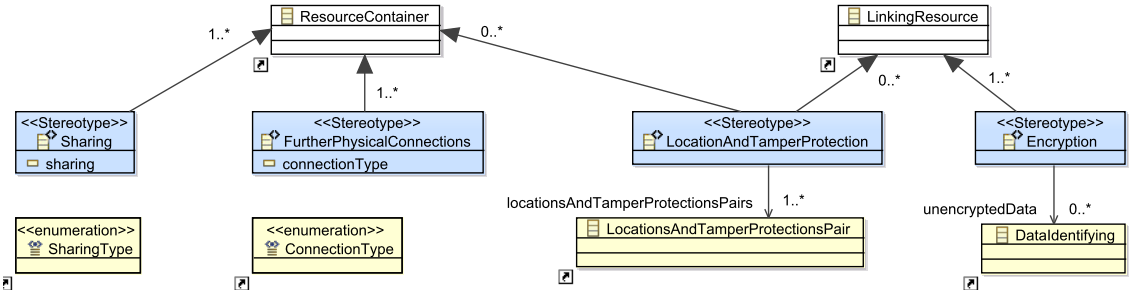

Abbildung 8: Referenzierung des UserIdentifiers

3.5 Adversary

Das Adversary Paket hat sich nur dadurch verändert, dass nur die Reference mayTamperWith nicht mehr auf die Klasse LocationAndTamperProtectionsPair zeigt, sondern auf die KlasseAbstractResourceProtection.

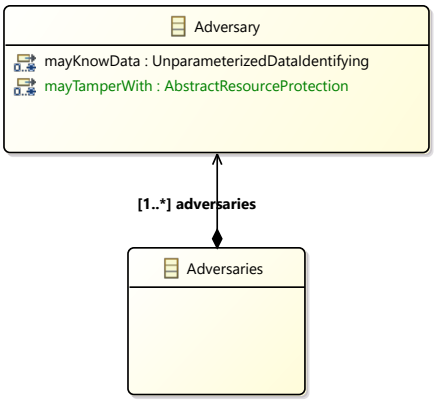

Abbildung 9: Referenzierung des UserIdentifiers

- Profile entfernt
- Genauere Modellierung des Informationsflusses
- Identifier aus PCM

4 PCM2Prolog Generator

Der PCM2Prolog Generator wandelt eine Instanz des Confientiality Modells in Prolog Prädikate um. Dieser musste im Rahmen dieses Projekts so angepasst werden, dass trotz des veränderten Modells der Prolog Code identisch ist.

Grundsätzlich verwendet der Generator die Reflective API von ECore und ist in xTend geschrieben. Für jedes Entität wird die Methode `String generateDeeply(EObject e)` aufgerufen. Diese generiert für jedes Object und dessen Kinder PrologPrädikate für alle Referenzen und Attribute. Der Form: `referenceName("objId", ["refObjId1","refObjId2"]).` In der Klasse *PCM2PrologXSBFilter* kann hierbei angegeben werden welche Klassen und Referenzen berücksichtigt werden sollen.

Für Objekte die nicht automatisch generiert werden können, können Dispatch-Methoden für die `String generateDeeply(EObject e)` Methode definiert werden. Dies wurde für viele der neu hinzugefügten Klassen gemacht, um sicherzustellen, dass die Prolog Prädikate identisch zum ursprünglichen Modell sind.

Durch das entfernen des Profil-Mechanismus hat sich die Richtung der Referenzen der PCM Komponenten zu den Veraulichkeitsinformationen umgedreht. Im ursprünglichen Modell hat ein Stereotype wie z.B. LocationAndTemperProtection angewandt auf einen ResourceContainer mehrere LocationAndTemperProtectionPairs und wird in ein einziges Prädikat umgewandelt. Im neuen Modell gibt es allerdings für jedes LocationAndTemperProtectionPair eine ResourceContainerProtection mit einer Referenz auf den enspechenden

ResourceContainer. Damit diese Referenzen trotzdem in ein Prädikat pro ResourceContainer umgesetzt werden können, werden die Referenzen in einer Map gespeichert wobei der Identifier des ResourceContainers als Key dient. So kann nach dem alle Objekte verarbeitet wurden, für jeden ResourceContainer in der Liste ein Prädikat generiert werden mit den entsprechenden Identifieren der zugehörigen ResourceContainerProtections. Das selbe Prinzip wird auch für die SpecificationParameterEquation, InformationFlowParameter und AbstractInformationFlows benötigt.

5 Evaluierung

Um zu überprüfuen ob der generierte Prolog Code identisch zu dem Prolog Code vor der Refaktorisierung ist wurden die Beispiel Projekte Cloud-minimzied[3] und IFlowExample[4] im neuen Modell modelliert und das Ergebnis mit dem Ergebnis des ursprünglichen Modells vergleichen. Diese zwei Beispiel Projekte wurden gewählt, da sie gemeinsam nahe zu alle Elemente des Confientiality Modells abdecken. Die einzige nicht verwendet Klasse ist die SpecificationParameter2DataSetAssignment Klasse. Diese wurde gesondert händisch getestet.
Um den Prolog Code automatisch vergleichbar zu machen müssen beim Modellieren die selben Identifier verwendet werden wie in original Projekt. Die Identifier der LocationAndTemperProtectionsPair Elemente müssen identisch zu den Identifierer der AbstractResourceProtection Elemente sein. Die Identifier der ParameterAndDataPair Elemente müssen identisch zu den Identifiern der Information Elemente sein. Die Identifier der anderen neuen Klassen können frei gewählt werden.

Beim generieren des Prolog Codes die Option fullIDs, singleFile und noComment ausgewählt werden, da bewusst die Kommentare nicht identisch sind, sondern so, dass auf die Entitäten aus denen die Prolog Prädikate generiert wurden zurück geschlossen werden kann.

Im letzten Schritt müssen die Ausgabe Dateien noch vereinheitlicht werden. Dies ist jediglich notwendig um die die Dateien automatisch vergleichen zu können. Es gibt viele Prädikate der Form:

```
prädikatName(”entity”,[”child1”,”child1”,”child3”]).
```

Die Reihenfolge der Elemente `child1` bis `child3` in der Liste ist hierbei irrelevant. Deshalb müssen die Listenelemente solcher Prädikate alphabetisch sortiert werden. Außerdem kann sich die Reihenfolge der Zeilen in unterscheiden. Deshalb wird anschließend die Datei zeilenweise sortiert und leere Zeilen entfernt. Diese beiden Schritte können mit einem einfachen Shell Skript durchgeführt werden. Auf den vereinheitlichen Dateien kann dann ein diff ausgeführt werden.

[3] https://github.com/KASTEL-SCBS/Examples4SCBS/tree/master/bundles/edu.kit.kastel.scbs.cloudscenario-minimized

[4] https://github.com/KASTEL-SCBS/Examples4SCBS/tree/master/bundles/edu.kit.kastel.scbs.iflowexample

Beide Beispiel Projekte CloudScenario-minimized udn IflowExample konnten erfolgreich so im neuen Modell modelliert werden, dass der Prolog Code identisch ist. Allerding mussten im CloudScenario-minimized Beispiel eine kleine Änderung vorgenommen werden, da eine Referenz auf ein LocationAndTemperProtectionsPair in cloud.adverary zwar defniniert war aber nicht im cloud.confidentielity File. In IFlowExample gab es undefinierte Referenzen innerhalb der Elementen *Allocation Energy Visualization*, *Allocation Database DBMS* und *Allocation Energy Meter* im File *default.allocation* und innerhalb der Elemente *ProvDelegation Provided AnInterface -> Provided AnInterface AComponent*, *Connector Energy Visualization -> Database* und *Connector Energy Visualization -> Energy Meter Assembly Context* und im *default.system* File. Da diese Elemente in in der Vertraulichkeitsanalyse referenziert werden, konnten diese gelöscht werden. Das Problem der undefinierten Referenzen ist, dass diese beim Generieren des Prolog Codes zu Prädikaten mit zufälligen Identifiern führen.