AML Helper

PSE “ASSISTENZTOOLS FÜR DEN MASCHINENBAU”

Pflichtenheft

Erstellt von

Max Baumstark, Ahmad Beidas, Dennis Brüstle,
Nadim Hammoud, Christopher Hommel und Kevin Kellner[1]

Betreuer:

Frau Dr.-Ing. Miriam Schleipen, Dr. Michael Okon, Robert Henßen[2]

30. November 2015

---

[1] Karlsruher Institut für Technologie

[2] Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung

Inhaltsverzeichnis

Einführung

Problemstellung

[1], [2], [3] Produzierende Unternehmen sind stets bemüht, Produkte höchster Qualität mit möglichst geringen Kosten anzubieten und hierbei möglichst hohe Produktionsraten zu erreichen, um einen hohen Profit einzufahren.
Produktionssysteme, das Produkt selbst, die Unternehmensverwaltung, Teile der Unternehmensinfrastruktur oder auch technologische und logistische Prozessen unterliegen einem konstanten Wandel und veralten. Dies stellt Unternehmen vor besondere Herausforderungen, da derartige Veränderungen, die oft zuerst beim Kunden oder Zulieferern auftreten, möglichst umgehend festgestellt und entsprechend adressiert werden sollten. Dies bedeutet nicht selten, Restrukturierungen in den zuvor genannten Bereichen des Unternehmens oder der Produktion zu implementieren.
Traditionell wird eine Änderung manuell durchgeführt, was in vielen Fällen zu einer Unterbrechung der Produktion und folglich einem Einbruch in den Einnahmen führt.

Lösungsansatz

Industrie 4.0! Durch cyber-physische Subsysteme, in denen Komponenten kommunizieren und unabhängige Entscheidungen treffen, kann das System zu einer autonomen Einheit werden, die Daten aus der Produktion sammelt, eine Echtzeitdarstellung des Status in der Produktion liefert und Fehler diagnostizieren kann. Eine “smart factory”, wie derartige Produktionsstätten bezeichnet werden, ist im Stande, Sektoren autonom zu modifizieren, ohne die übrigen Einheiten zu berühren.

Tatsächlich zeigt eine Untersuchung von AIDI, dass ein großer Kostenanteil für die Automatisierung einer Produktion durch hohe Datenübertragungskosten bestimmt wird. Hierbei ist vor allem ausschlaggebend, dass verschiedene Systeme bei der Verarbeitung und Speicherung von Informationen zum Einsatz kommen, weshalb oft eine manuelle, ineffiziente Datenübertragung nötig ist. Führende Unternehmen, darunter Daimler, ABB, KUKA, Rockwell und Siemens haben in Zusammenarbeit mit dem Karlsruher Institut für Technologie (KIT) sowie der Universität Magdeburg folgerichtig begonnen, einen einheitlichen Standard (AutomationML, kurz AML) für den nahtlosen und damit automatischen Austausch zu entwickeln. AutomationML soll universell im gesamten Produktionsprozess einsetzbar sein und es möglich machen, diverse Informationen, etwa Anlagenbeschreibungen, zu speichern und somit komplexe Produktionen und Systeme zu modellieren.

Gegenstand dieser Arbeit

Um AML zur Verbreitung zu verhelfen, soll die Erstellung von AML-Dateien möglichst einfach vonstattengehen. Insbesondere Ingenieure ohne informationstechnischen Hintergrund sollen ohne aufwändige Einarbeitung in der Lage sein, AML-Dateien zu erstellen. Deshalb liefert der für die Entwicklung von AML zuständige Verein einen offiziellen AML-Editor, um die grafische Erstellung und Bearbeitung von AML-Dateien zu ermöglichen. Unglücklicherweise ist die aktuelle Oberfläche des Editors - insbesondere für neue oder fachfremde Benutzer und bei Arbeiten an komplexeren Modellen - oft unübersichtlich und komplex und damit wenig attraktiv. Um diesem Mangel entgegen zu wirken und die Arbeit mit dem AML-Editor zu vereinfachen als auch attraktiver zu gestalten, soll ein Plugin mit einer fortgeschrittenen, intuitiven Oberfläche und Benutzerführung entwickelt werden. Damit soll ein Beitrag zur Etablierung des AutomationML-Formats geleistet werden.

1. Zielbestimmung

Benutzer des AutomationML-Editors (AML-Editor) sollen durch das zu erstellende Plugin in die Lage versetzt werden, AML-Dateien einfacher und effizienter zu erstellen und zu manipulieren.

1.1 Musskriterien

Erstellung neuer, valider AML-Dateien
Veränderung und Anzeige bereits bestehender AML-Dateien
Fortgeschrittene, grafische Benutzeroberfläche
Intuitive, internetbrowserähnliche Bedienung
Nutzung des Plugins mit dem AML-Editor

1.2 Wunschkriterien

Unterstützung mehrerer Sprachen
Umfassende Tastaturnavigation
Änderungsverlauf speichern und Zurückgehen ermöglichen
Erweiterte Filter- und Suchfunktionen
Sichern der aktuellen Arbeitsoberfläche in einer Projektdatei
Einführungs- und Hilfefunktionen
Umfangreichere Einstellungsmöglichkeiten
“Rückgängig”-Funktion

1.3 Abgrenzungskriterien

Mit dem Plugin soll es möglich sein, konkrete Modelle zu erstellen. Die Erstellung oder Manipulation von Bibliotheken wird hingegen nicht unterstützt.

2. Produkteinsatz

Das Produkt soll von Nutzern des AutomationML-Editors zur vereinfachten Erstellung und Manipulation von AML-Dateien eingesetzt werden.

2.1 Anwendungsbereiche

Einsatz bei AutomationML als Erweiterung des AutomationML-Editors

2.2 Zielgruppen

Alle Nutzer des AutomationML-Editors, vornehmlich Planer von Produktionsanlagen, grundsätzlich aber offen

2.3 Betriebsbedingungen

Büroumgebung, Firmenumgebung

3. Produktumgebung

Der Einsatz erfolgt auf einem üblichen Arbeitsplatzrechner oder einem ähnlichen mobilen Rechner.

3.1 Software

AutomationML-Editor 4.1.6.0, AutomationML-Engine 3.1
Betriebssystem: Windows 7 oder höher (Keine Unterstützung für OS X bzw. Linux)

3.2 Hardware

Im Jahre 2015 gängiger Arbeitsplatzrechner oder mobiler Rechner, für genaue Anforderungen siehe Mindestvoraussetzungen des AutomationML-Editors.
Empfohlene Hardware: mindestens 4GB RAM, 2 GHz Intel i3 oder besser

3.3 Produktschnittstellen

Das Plugin kommuniziert mit dem AutomationML-Editor über dessen Plugin-Schnittstellen und verwendet die AutomationML Engine. Die erstellten AML-Dateien können mit dem AutomationML-Editor geöffnet und bearbeitet werden und sollen wie vorgesehen eingesetzt werden.

4. Funktionale Anforderungen

4.1 Dateibezogene Funktionalität

/FA 100/

Ausschließlich AML-Bibliotheken, die vom AML Editor akzeptiert werden, können über die “Öffnen”-Funktion im Menü geladen werden.

/FA 110/

Ausschließlich AML-Dateien, die vom AML Editor akzeptiert werden, können über die “Öffnen”-Funktion im Menü geladen werden.

/FA 110.1/

Dateien, die nicht die Dateiendung .aml tragen, werden in der “Öffnen”-Funktion ausgeblendet.

/FA 120/

Über die “Speichern”-Funktion im Menü kann die aktuell bearbeitete AML-Datei an einem auswählbaren Speicherort gespeichert werden.

/FA 130/

Über die “Neu”-Funktion im Menü kann eine neue AML-Datei erstellt werden.

/FA 140/

Die aktuell geöffnete Datei kann über die “Datei schließen”-Funktion im Menü geschlossen werden.

4.2 Pluginfenster

/FA 200/

Das Plugin öffnet sich in einem neuen Fenster als schreibendes Plugin (stand alone, editing plugin).

/FA 210/

Das Plugin besitzt eine vom AutomationML-Editor unabhängige Schließ-, Minimier- und Maximierfunktion.

4.3 Speicherabfragen, Warnungen und Meldungen

/FA 300/

Beim Schließen des Plugins soll, sofern eine Datei geöffnet ist und verändert wurde, eine Abfrage erfolgen, ob die Datei gespeichert werden soll.

/FA 310/

Ist bereits eine AML-Datei mit ungespeicherten Änderungen geladen und soll eine neue Datei angelegt oder geöffnet werden, so erfolgt eine Nachfrage, ob die aktuell bearbeitete Datei gespeichert werden soll.

/FA 320/

Beim Speichern von AML-Dateien, die vom AML Editor als fehlerhaft eingestuft werden, wird eine Warnung angezeigt.

/FA 330/

Neu erstellte Elemente, bei denen alle Werte noch den Standardwerten entsprechen, werden durch ein Symbol gekennzeichnet.

/FA 340/

Beim Löschen von Elementen wird eine Warnung angezeigt und die Operation muss bestätigt werden

/FA 350/

Warnungen lassen sich individuell ausschalten.

4.4 Registerkarten

/FA 400/

Es ist möglich, Elemente in Registerkarten zu öffnen und zwischen diesen per Mausklick zu wechseln.

/FA 410/

Das zuvor in einer Registerkarte angezeigte Element kann erneut in ebenjener Registerkarte angezeigt werden, vergleichbar mit einer “Zurück”-Funktion in einem Browser.

/FA 420/

Eine “Vorwärts”-Funktion für die Registerkarten, vergleichbar mit der eines Browsers.

/FA 430/

Jedes Element kann in der aktuellen Registerkarte oder in einer neuen Registerkarte geöffnet werden, vergleichbar mit Links eines Browsers.

/FA 440/

Registerkarten können in der Registerkartenleiste verschoben und so neu angeordnet werden.

/FA 450/

Geöffnete Registerkarten können per Mausklick geschlossen werden.

/FA 460/

Alle Registerkarten können mit einer Aktion gleichzeitig geschlossen werden.

/FA 470/

Alle Registerkarten bis auf eine können gleichzeitig geschlossen werden.

4.5 Darstellung von Elementen und Attributen

/FA 500/

Ist ein Element in einer Registerkarte geöffnet, können alle ihm untergeordneten Elemente aufgeteilt in Typgruppen angezeigt werden.

/FA 500.1/

Offene Typgruppen lassen sich innerhalb einer Registerkarte einzeln verstecken.

/FA 500.2/

Typgruppen lassen sich innerhalb einer Registerkarte einzeln anspringen.

/FA 510/

Ist ein Element in einer Registerkarte geöffnet, können alle seine Attribute einschließlich ihres Namens, ihrer Beschreibung, ihrem Wert, des Standardwerts, der Einheit und des Datentyps angezeigt werden.

/FA 510.1/

Die Darstellung von Attributen innerhalb einer Registerkarte lässt sich verstecken.

/FA 510.2/

Zur Darstellung der Attribute innerhalb einer Registerkarte kann gesprungen werden.

/FA 520/

Wenn ein Element in einer Registerkarte geöffnet ist, wird das ihm übergeordnete Element, sofern eines vorhanden ist, angezeigt.

/FA 530/

Elemente können durch den ihnen vergebenen Namen auf den Registerkarten oder in der Wurzelansicht identifiziert werden.

4.6 Darstellung und Funktionen der Hierarchie

/FA 600/

Es existiert eine vollständige, hierarchische Darstellung der Strukturen der aktuell bearbeiteten AML-Datei.

/FA 610/

In der Hierarchie können die Unterelemente eines Elements ein- und ausgeblendet werden.

/FA 620/

Elemente können direkt aus der Hierarchie heraus in einer Registerkarte geöffnet werden.

/FA 630/

Die Reihenfolge der Module kann per “Ziehen und Ablegen” verändert werden, solange die Hierarchie nicht verletzt wird.

/FA 640/

Alle Unterelemente eines Elements in der Hierarchiedarstellung können mit einer Aktion ein- oder ausgeklappt werden.

/FA 650/

Mithilfe einer Filterfunktion können Elemente der Hierarchie gesucht werden.

4.7 Elemente und AML-Beziehungen

/FA 700/

Elemente können angelegt und bearbeitet werden.

/FA 710/

Elemente können gelöscht, kopiert und ausgeschnitten werden.

/FA 720/

Bei einem Attribut können der Name, die Beschreibung, der Wert, der Standardwert, die Einheit und der Datentyp verändert werden.

/FA 740/

Zu einem InternalElement können Elemente hinzugefügt werden und zwar ausschließlich wie folgend spezifiziert, wobei die Nennung der Einzahl eine Mehrzahl nicht ausschließt:

/FA 740.3/

Es kann eine oder mehrere SupportedRoleClass ausgewählt werden.

/FA 740.4/

Es kann ein InternalLink hinzugefügt werden.

/FA 750/

Zu einer SupportedRoleClass können ein oder mehrere MappingObjects hinzugefügt werden.

/FA 760/

Zu einem MappingObject können AttributeNameMapping hinzugefügt werden.

/FA 760.1/

Zu einem MappingObject können InterfaceNameMapping hinzugefügt werden.

/FA 770/

Zu einer AML-Datei können eine InstanceHierarchy oder mehrere hinzugefügt werden.

/FA 780/

Zu einer InstanceHierarchy kann ein InternalElement oder mehrere hinzugefügt werden.

4.8 AML Editor

/FA 800/

Das Plugin soll dem AML Editor als Plugin hinzugefügt werden können und mit diesem dann verwendbar sein.

4.9 Konsistenzen

/FA 900/

Die Darstellungen (Registerkarten und Hierarchie) und Funktionen (bspw. Rückgängigmachen oder Element löschen) müssen automatisch auf gegenseitige Änderungen adäquat reagieren, um die Konsistenz zu wahren.

5. Produktdaten

5.1 Temporäre Daten

/PD 100/

Für jede Registerkarte werden die zuvor in dieser Registerkarte geöffneten Elemente chronologisch gespeichert.

/PD 110/

Beim Verwenden der Zurückfunktion in einer Registerkarte, wird das Element gespeichert, das zuvor in der Registerkarte geöffnet war.

/PD 120/

Änderungen werden als Delta gespeichert.

/PD 130/

Zuletzt geschlossene Registerkarten und deren Einstellung vor dem Schließen werden gespeichert.

5.2 Projektdatei

/PD 200/

Die geöffneten Registerkarten und ihre Einstellungen werden beim Schließen des Projekts gespeichert.

/PD 210/

Der Status der Hierarchiedarstellung wird beim Schließen des Projekts gespeichert.

5.3 Konfigurationsdatei

/PD 300/

Vorgenommene Einstellungen, die das Plugin generell betreffen, etwa Spracheinstellungen, sollen beim Schließen des Plugins gespeichert werden.

5.4 AML-Datei

/PD 400/

Das AML-Modell wird auf Anforderung des Benutzers oder, wenn gewünscht, beim Schließen gespeichert.

6. Nichtfunktionale Anforderungen

/NF 100/ Reaktionszeit

Nach einer Aktion gibt es innerhalb von einer Sekunde eine Rückmeldung.

/NF 200/ Registerkartenzahl

Es müssen bis zu 10 Registerkarten gleichzeitig geöffnet sein können.

/NF 300/ Navigationsschritte

Es muss möglich sein, bis zu 10 Navigationsschritte in einer Registerkarte zurückzugehen.

/NF 300.1/

Es muss möglich sein, bis zu 10 Navigationsschritte in einer Registerkarte vorzugehen.

/NF 400/ Registerkarten wiederherstellen

Es müssen bis zu 10 geschlossene Registerkarten wiederhergestellt werden können.

/NF 500/ Modellumfang

Es müssen AML-Modelle mit einem Umfang von insgesamt bis zu 500 Internen Elementen gehandhabt werden können.

/NF 600/ Fehlertoleranz

Es sollen Inkonsistenzen mit der Projektdatei vermieden, erkannt und entsprechend behandelt werden.

/NF 700/ Benutzbarkeit

Das Plugin soll nach einer kurzen Einführung (< 1 Stunde) selbstständig benutzt werden können. Die Benutzung soll intuitiv sein, die grafische Oberfläche fortgeschritten und attraktiv.

/NF 800/ Zuverlässigkeit

Das Plugin soll, sofern es in den spezifizierten Rahmenbedingungen korrekt verwendet wird, bei 1000 Einsätzen nicht häufiger als einmal unerwartet nicht mehr funktionieren.

/NF 900/ Übertragbarkeit

Die mit dem Plugin erstellten AML-Dateien sind mit dem AutomationML Editor benutzbar.

/NF 1000/ Richtigkeit

Es ist möglich, korrekte AML-Dateien zu erstellen.

/NF 1100/ Interoperabilität

Das Plugin arbeitet ordnungsgemäß mit dem AML Editor zusammen.

/NF 1200/ Testbarkeit

Das Plugin soll mit den üblichen Testmitteln geprüft werden können.

/NF 1300/ Wartbarkeit

Einfachere Oberflächenänderungen sollen effizient möglich sein.

7. Globale Testfälle

7.1 Testfälle für funktionale Anforderungen

/TFA 100/

Das Öffnen, Verändern, Abspeichern und anschließende Schließen einer *.aml Datei erfolgt problemlos.

/TFA 100.1/

Es wird versucht, eine Datei zu öffnen, die nicht die Endung .aml trägt. Die Datei wird dabei im “Datei öffnen”-Dialog nicht angezeigt.

/TFA 100.2/

Es wird versucht, eine Datei zu öffnen, die nicht vom AML Editor akzeptiert wird. Das Öffnen schlägt fehl.

/TFA 100.3/

Es wird versucht, eine gültige AML-Bibliothek zu öffnen. Diese wird erfolgreich geladen.

/TFA 100.4/

Es wird versucht, eine Bibliothek zu öffnen, die nicht vom AML Editor akzeptiert wird. Dies schlägt fehl.

/TFA 200/

Das Plugin wird dem AML Editor erfolgreich hinzugefügt und in diesem sodann gestartet.

/TFA 300/

Registerkartennavigation: Schließen, leere Registerkarte (RK) öffnen, RK mit Inhalt öffnen und verschieben.

/TFA 300.1/

Neue Registerkarten durch (doppel-)klicken auf den Namen eines Elements, entweder in der Hierarchie oder in einer Liste, öffnen

/TFA 300.2/

Es wird in einer Registerkarte ein Element angeklickt, sodass dieses nun in der Registerkarte angezeigt wird. Nun kann über die “Zurück”-Funktion das zuvor geöffnete Element angezeigt werden.

/TFA 300.3/

Es wird je ein Element jeden Typs in einer Registerkarte geöffnet. Dabei ist die Darstellung des Elements jeweils korrekt und vollständig.

/TFA 400/

Alle beschriebenen AML-Beziehungen lassen sich modellieren.

/TFA 500/

Die Filterfunktion schränkt die angezeigten Elemente ein. Nach dem Löschen des Suchbegriffs wird der ursprüngliche Baum angezeigt.

/TFA 600/

In der Projektdatei ist der Status des Programms (Hierarchie, geöffnete Registerkarten) gespeichert, nachdem diese angelegt wurde.

/TFA 700/

Konsistenz zwischen den Inhalten der Registerkarten und der Hierarchie wird geprüft. Hierzu werden in der Hierarchie Veränderungen vorgenommen und es wird getestet, ob die Registerkarten angemessen reagieren. Daraufhin werden in den Registerkarten Änderungen durchgeführt und eine angemessene Reaktion der Hierarchie wird überprüft.

/TFA 800/

Ein Panel fügt nur Elemente des zugehörigen Typs ein (z.B. das Panel für Interne Elemente fügt keine Schnittstelle ein).

/TFA 900/

Die Hierarchiedarstellung ist vollständig und korrekt. Sie bleibt es, nachdem Elemente hinzugefügt und daraufhin gelöscht werden.

/TFA 900.1/

Aus der Hierarchiedarstellung heraus wird ein Element in einer neuen Registerkarte geöffnet. In der Registerkarte öffnet sich das korrekte Element.

/TFA 1000/

Es werden Einstellungen vorgenommen und diese gespeichert. Die nun entstehende Konfigurationsdatei enthält die korrekten Einstellungen.

/TFA 1100/

Es wird eine korrekte AML-Datei geöffnet, gültig verändert und versucht, das Plugin zu schließen, ohne die Datei vorher zu speichern. Sofern die Warnungen nicht ausgeschaltet wurden, erfolgt eine Warnung.

7.2 Testfälle für nichtfunktionale Anforderungen

/TNFA 100/

Nach dem Öffnen einer Datei gibt es innerhalb von einer Sekunde eine Rückmeldung.

/TNFA 200/

Es werden 10 Registerkarten geöffnet. Das Programm läuft stabil.

/TNFA 300/

Es wird ein AML-Modelle mit einem Umfang von 500 Elementen geöffnet. Das Programm läuft stabil.

/TNFA 500/

Es werden 10 Registerkarten geöffnet und dann nacheinander geschlossen. Alle drei Registerkarten lassen sich sodann exakt wiederherstellen.

/TNFA 600/

Eine Registerkarte mit einem Element wird geöffnet. Es werden 10 Navigationsschritte in dieser Registerkarte nach vorne gemacht. Dann werden 10 Navigationsschritte in derselben Registerkarte zurück gemacht; zwischendurch erfolgen keine weiteren Aktionen. Nun wird wieder dasselbe Elemente wie vor dem ersten Navigationsschritte nach vorne angezeigt.

/TNFA 700/

Es wird ein gültiges AML-Modell geöffnet. Es werden dann mehrere Elemente in verschiedenen Registerkarten geöffnet. Dann wird das Plugin mit Abspeichern einer Projektdatei geschlossen. Nun wird dasselbe AML-Modell manuell so geändert, dass mindestens ein Element, das in einer Registerkarte geöffnet war, entfernt wird. Die Änderung ist gültig, so dass weiterhin ein valides AML-Modell vorliegt. Nun wird das AML-Modell samt zugehöriger Projektdatei geöffnet. Das Plugin registriert die Entfernung und schließt die betreffende Registerkarte.

/TNFA 800/

Es wird eine Befragung mit mindestens 10 Personen durchgeführt, die nicht direkt an der Entwicklung des Plugins beteiligt sind. Die Personen werden hinsichtlich Attraktivität der

Benutzeroberfläche und intuitiver Bedienung befragt und geben eine Bewertung ab. Es kann auch ein Vergleich mit der bestehenden Oberfläche erfolgen.

/TNFA 900/

Es soll die Position der Filterfunktion der Hierarchie durch das Entwicklerteam verändert werden. Die dafür benötigte Zeit wird gemessen und sollte unter 1 Stunde liegen.

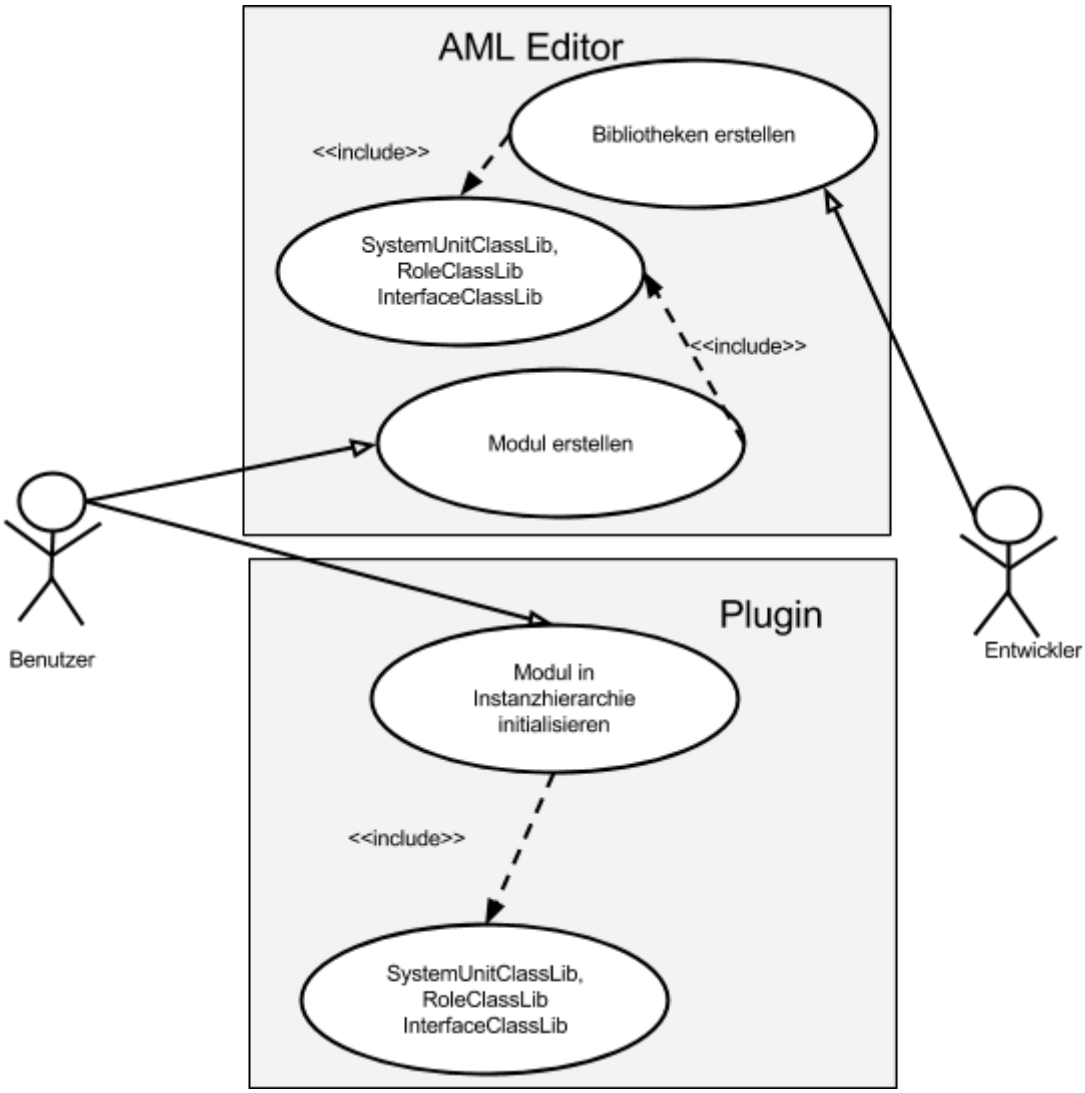
8. Systemmodelle

8.1 Szenario (gestützt auf „SecPnW_Industriewaschmaschine.aml“)

Martin möchte eine neue Industriewaschanlage des Typs Dolphin modellieren.
Er öffnet den AutomationML Editor 4.1.6 und startet das Plugin. Martin entscheidet sich dafür, die AML-Datei „SecPnW_Industriewaschmaschine.aml“ über *Datei->Öffnen* zu öffnen und darin eine neue Anlage zu erstellen. Er nennt die neue Anlage „Dolphin“ (vgl. Abb. 1, Kap 8.3), indem er diesen Namen in das dafür vorgesehene Feld einträgt.
Anschließend erstellt er ein neues InternalElement „Waschen“, welches in der Baumhierachie auf der linken Seite hinzugefügt wird und in der aktuellen Registerkarte der Anlage in der Typgruppe „Existierende InternalElements“ angezeigt wird.
Um schneller zwischen den Elementen wechseln zu können, öffnet er das neue InternalElement „Waschen“ über einen Rechtsklick auf das Element im Baum in einer neuen Registerkarte und wechselt in diese. Nun werden ihm Optionen und Informationen über dieses Element angezeigt (vgl. Abb. 2, Kap 8.3., hier mit allen Unterelementen geöffnet).
Will er nun die in diesem Element untergeordneten Interfaces einsehen oder ein neues Interface hinzufügen, so kann er diese durch Anklicken aufklappen (in Abb. 2 Punkt 2, Kap 8.3 bereits offen). Auf diese Weise kann er die Subelemente auch wieder verbergen.
Martin entscheidet sich dafür, ein neues Interface „NachfolgerModulInterface“ zu dem InternalElement „Waschen“ hinzuzufügen und öffnet es durch einen Klick darauf in derselben Registerkarte. Nun kann er sowohl den Namen einsehen und ändern, als auch die Attribute des Interfaces auflisten lassen(vgl. Abb. 3, Kap 8.3). Zugunsten der Übersichtlichkeit kann die Attributsliste auch ausgeblendet werden.

Nach dem Erstellen weiterer Elemente kann Martin seine neu erstellte Anlage in der Baumhierachie auf der linken Seite der grafischen Benutzeroberfläche betrachten und einfach durch sie navigieren (vgl. Abb.1 Punkt 6, Kap 8.3).
Martin ist mit seiner Arbeit zufrieden und speichert diese über *Datei->Speichern unter* mit einem geeigneten Namen an gewünschter Stelle ab.

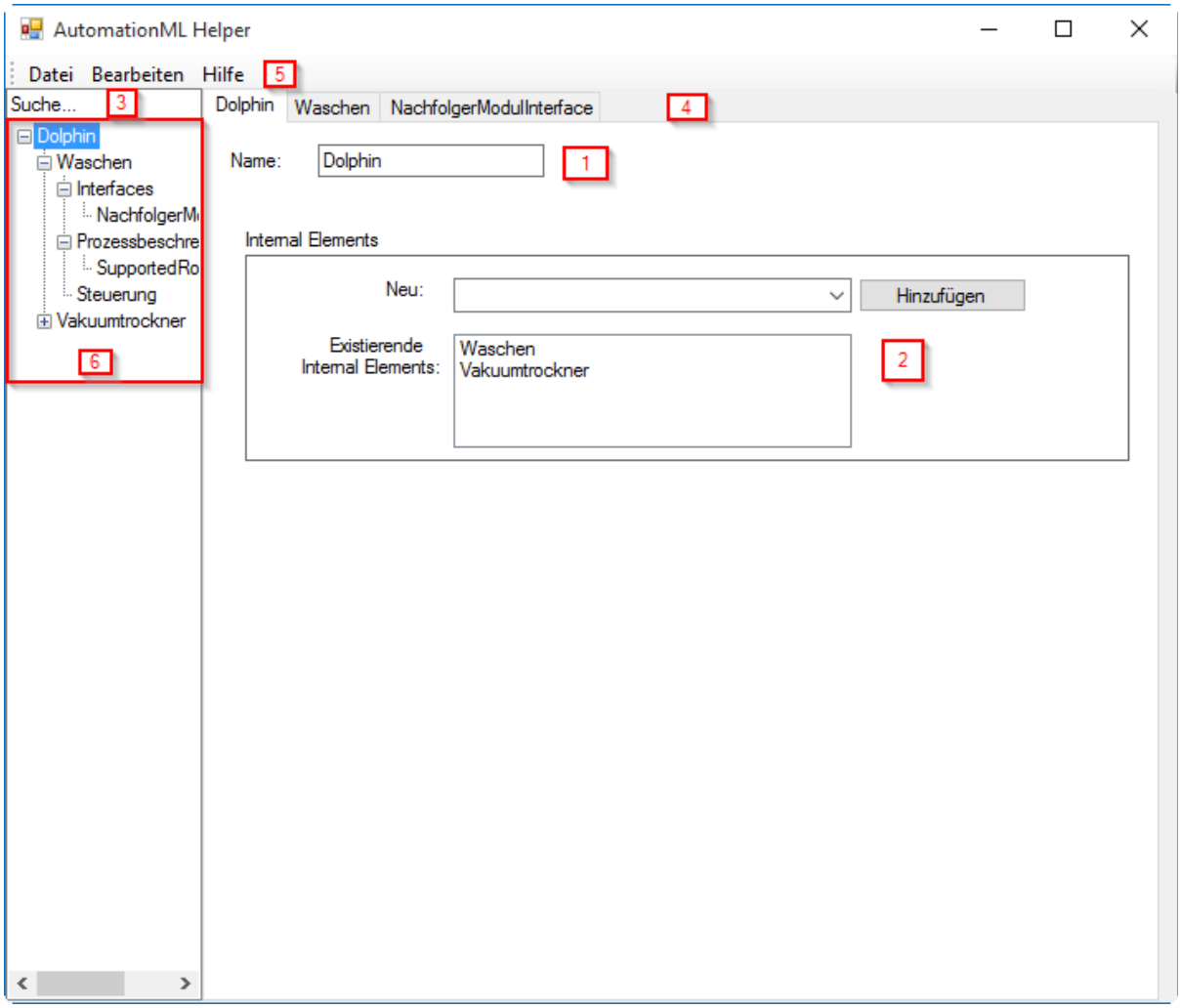
8.2 Anwendungsfalldiagramm



(Abbildung 0: Anwendungsfalldiagramm)

Entwickler können mithilfe des AML Editors Bibliotheken erstellen. Darunter fallen die SystemUnitClassLib, RoleClassLib und die InterfaceClassLib. Auch dem Benutzer ist es möglich, mithilfe des AML Editors neue Module zu erstellen.
Mit dem Plugin ist es nur möglich, neue Module zu erstellen, wobei sich auf die Bibliotheken gestützt wird, die vorher von dem Entwickler erstellt worden sind.
Außerdem ist sind die Funktionen des Plugins lediglich eine Teilmenge der Funktionen des AML Editors, da Module nicht völlig frei erstellt, sondern nur initialisiert werden können.

8.3 Skizzen der grafischen Benutzeroberfläche



(Abbildung 1: Skizze des Plugin Fensters, geöffnete Instanz-Registerkarte)

Im Textfeld (1) am oberen Rand des Panels kann der Name der InstanceHierarchy festgelegt werden.
Unterhalb des Textfelds befindet sich das “InternalElements”-Feld, in dem neue InternalElements aus einer Liste hinzugefügt werden können. In der Liste befinden sich alle Elemente der SystemUnitClassLib.

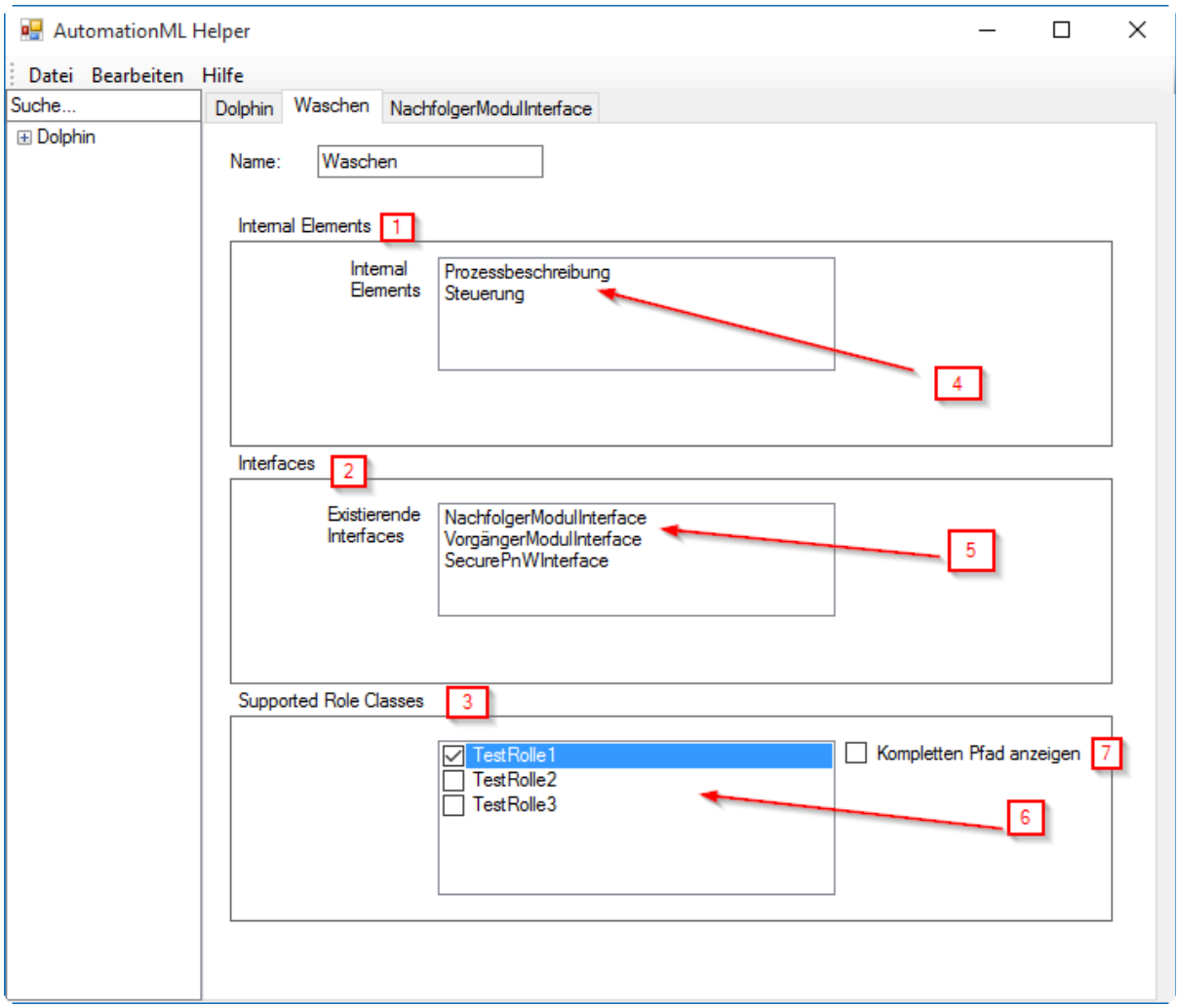
In der Liste darunter (2) werden alle InternalElements angezeigt, die sich in der Hierarchie eine Ebene unterhalb der Instanz befinden.

In dem Textfeld (3), das sich oberhalb der Baumstruktur (6) befindet, kann man die derzeitige Hierarchie mit einem Suchbegriff filtern lassen.

In der Registerkartenleiste (4) werden alle derzeitig geöffneten Registerkarten angezeigt. Nur InstanceHierarchies, InternalElements oder Interfaces lassen sich als Registerkarte öffnen. Die Registerkarten lassen sich schließen und umordnen.

Sind zu viele Registerkarten geöffnet, so erscheint ein Überlaufmenü am rechten Rand, in dem die restlichen Registerkarten gelistet sind.

In der Menüleiste (5) befinden sich die Menüpunkte “Datei”, “Bearbeiten” und “Hilfe”. Im “Datei”-Menüpunkt lassen sich neue Bibliotheken laden, das aktuelle Projekt speichern, oder ein neues Projekt anlegen.
Im “Bearbeiten”-Dialog befindet sich der “Rückgängig”-Menüpunkt, der es erlaubt, die letzte Aktion rückgängig zu machen.

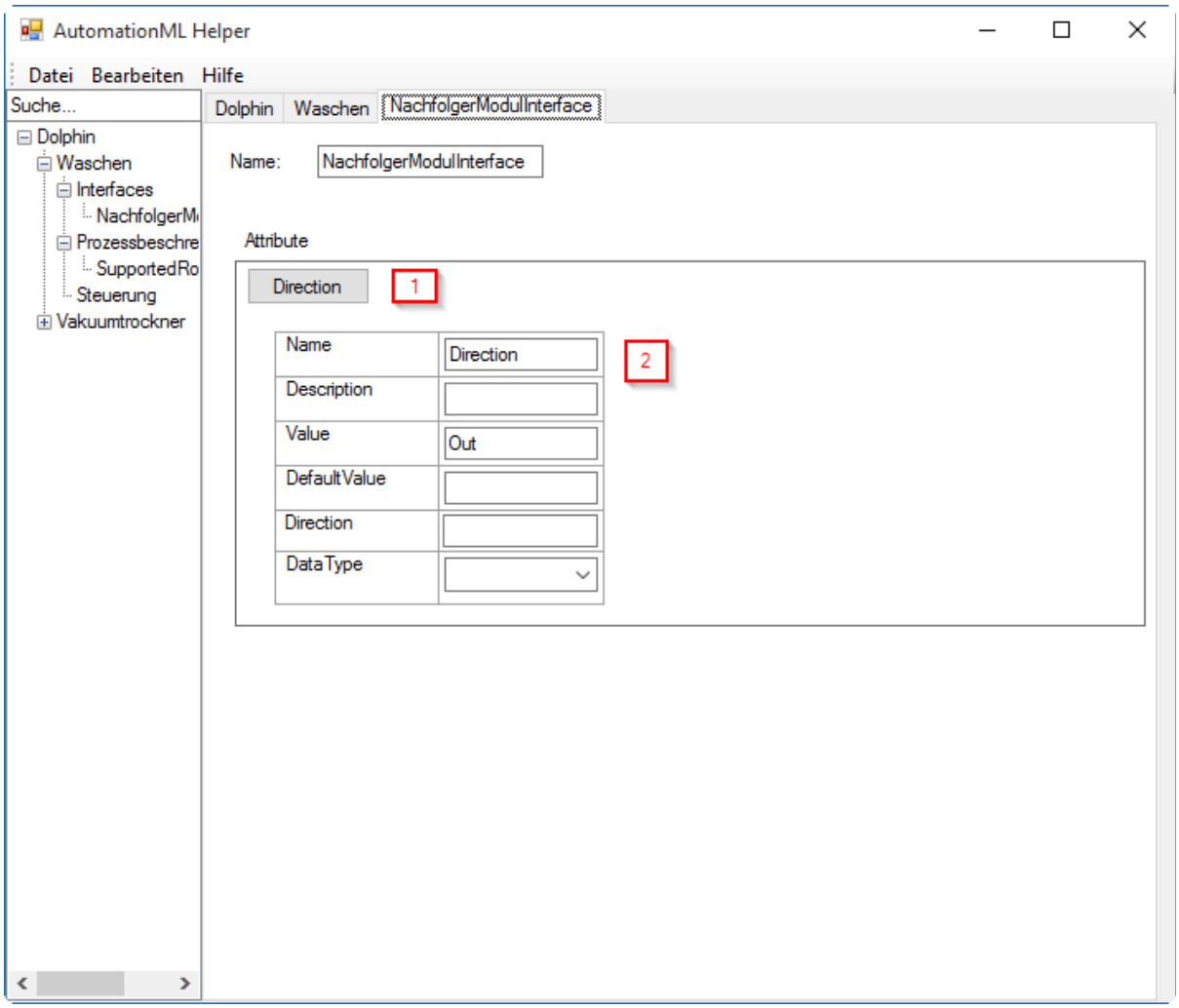


(Abbildung 2: Skizze des Plugin Fensters, geöffnete “InternalElement”-Registerkarte)

Eine “InternalElement”-Registerkarte ist in 4 Bereiche unterteilt (hier nur 3 sichtbar). Der erste (1) ist ähnlich zum “InternalElements”-Bereich bei der “InstanceHierarchy”-Registerkarte, nur ist es hier nicht möglich, Interne Elemente hinzuzufügen

Im Interface-Bereich (2) werden alle Interfaces des Moduls angezeigt.

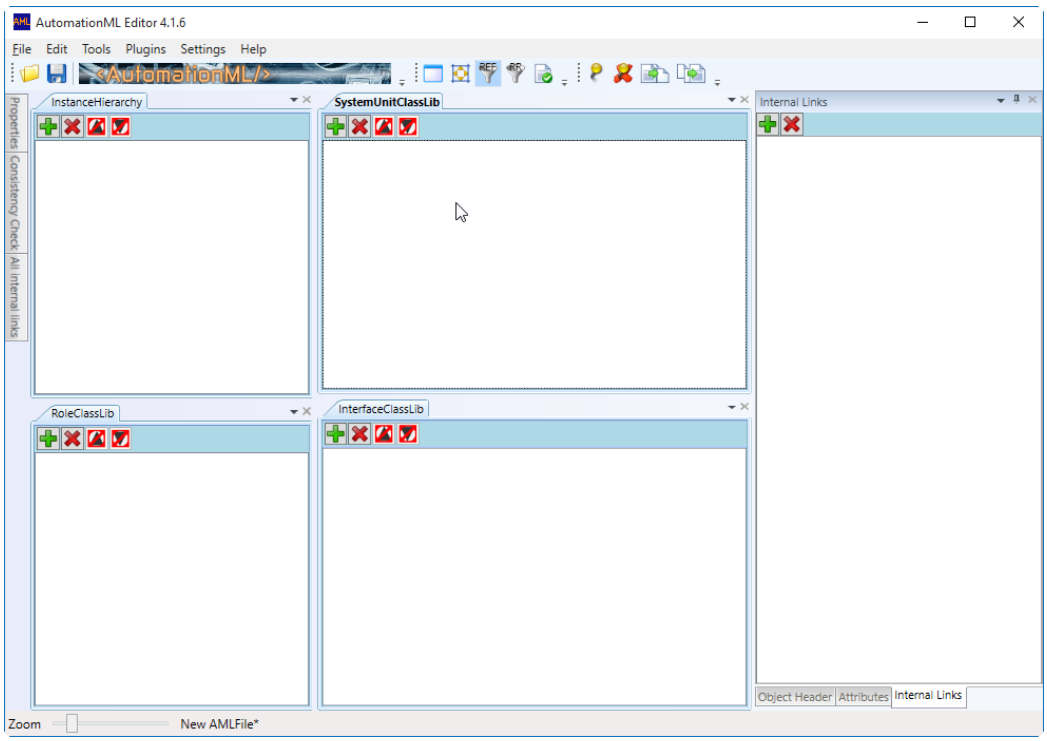
Im “SupportedRoleClasses”-Bereich (3) befinden sich alle SupportedRolleClasses des InternalElements. Hier werden über Kontrollkästchen die gewünschten Unterstützten Rollen ausgewählt.
Neben der Liste befindet sich ein weiteres Kontrollkästchen, mit dem eingestellt werden kann, ob der komplette Pfad einer SupportedRoleClass (innerhalb der SupportedRoleClassLib) angezeigt werden soll. Der vierte (hier nicht sichtbare) Bereich ist der Attributbereich, der über eine Bildlaufleiste erreichbar ist. Dieser wird nachfolgend beschrieben.



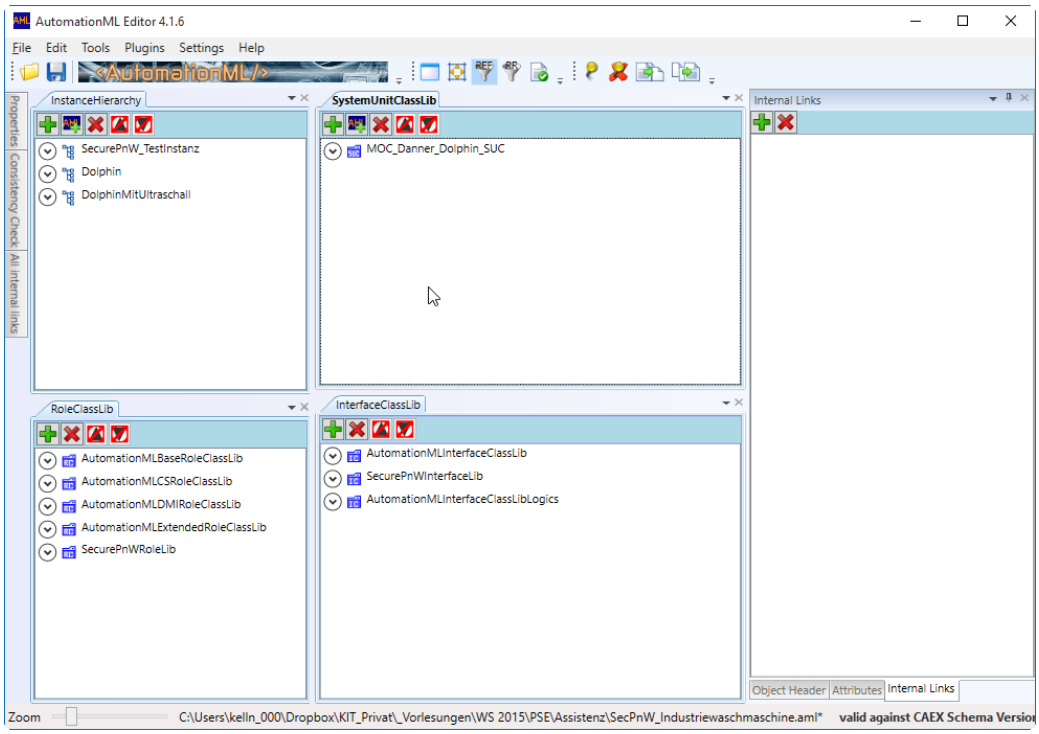
(Abbildung 3: Skizze des Pluginfensters: Geöffnete “Interface”-Registerkarte)

In der “Interface”-Registerkarte befindet sich der Attribute-Bereich. Jedes Attribut hat einen Namen (1). Klickt man auf den Namen, minimiert sich das Attribut inklusive der Tabelle (2). In der Tabelle befinden sich 6 Textfelder, mit denen die Werte eines Attributs gesetzt werden können.

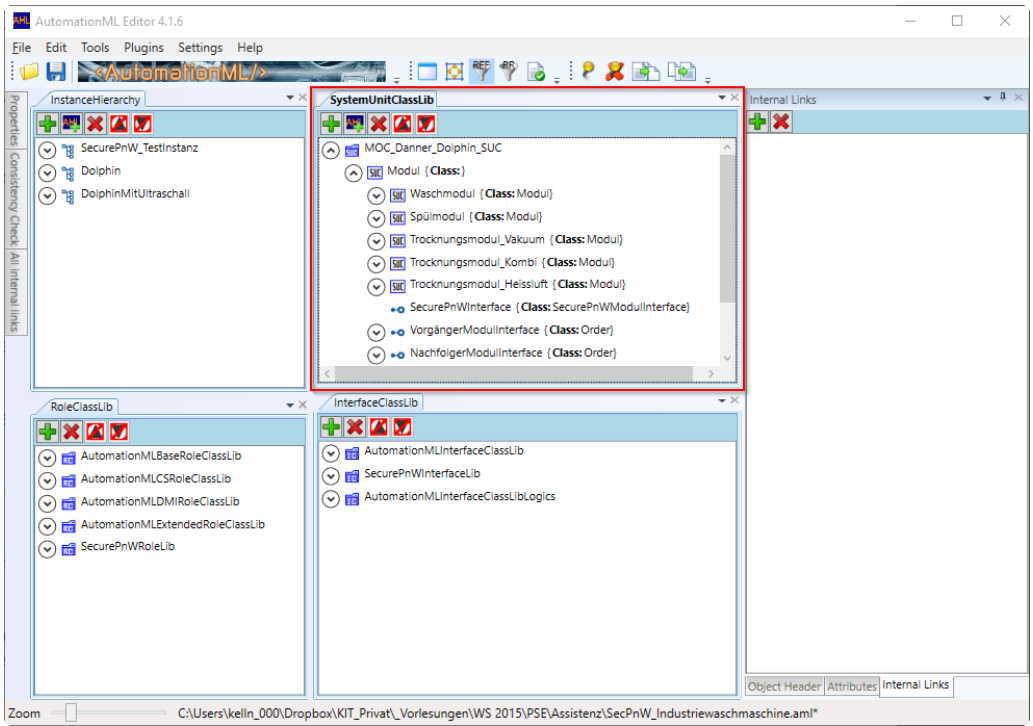
9. Anhang



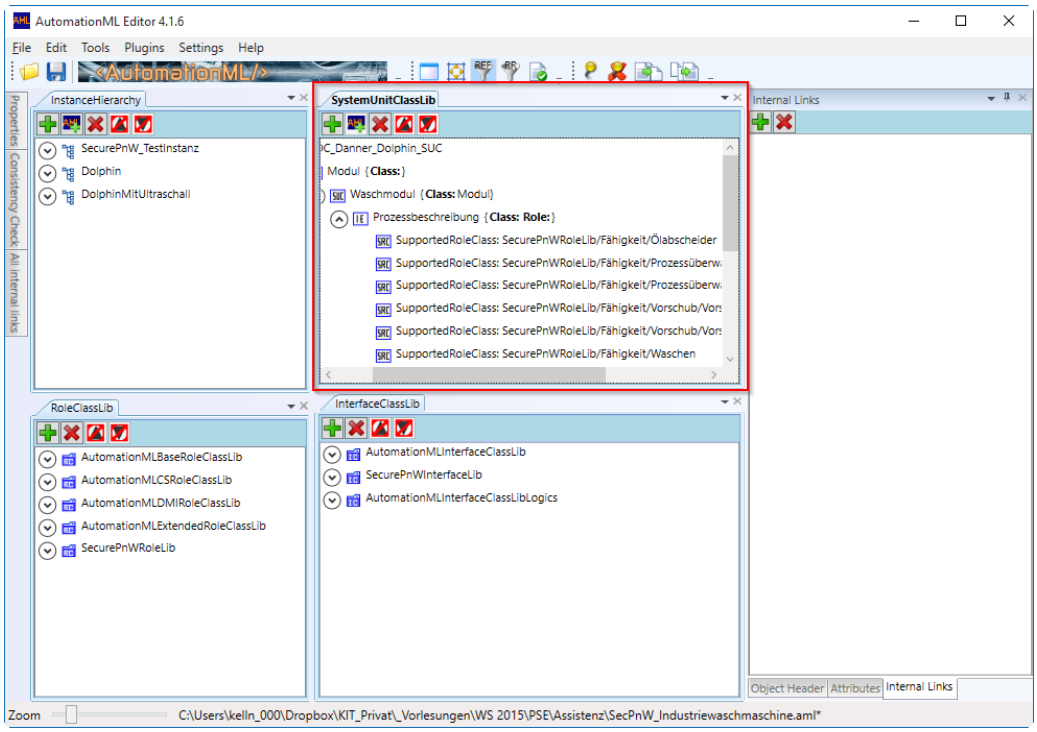
(Abbildung 4: Fenster des AML Editors)
Das Standardfenster beim Öffnen des AML Editors. Es ist in 5 Teile aufgetrennt, die im Folgenden erklärt werden.



(Abbildung 5: AML Editor mit geladener AML Datei)
Nach dem Laden einer AML-Datei füllen sich die InstanceHierarchie, SystemUnitClassLib, RoleClassLib und die InterfaceLib mit Elementen.

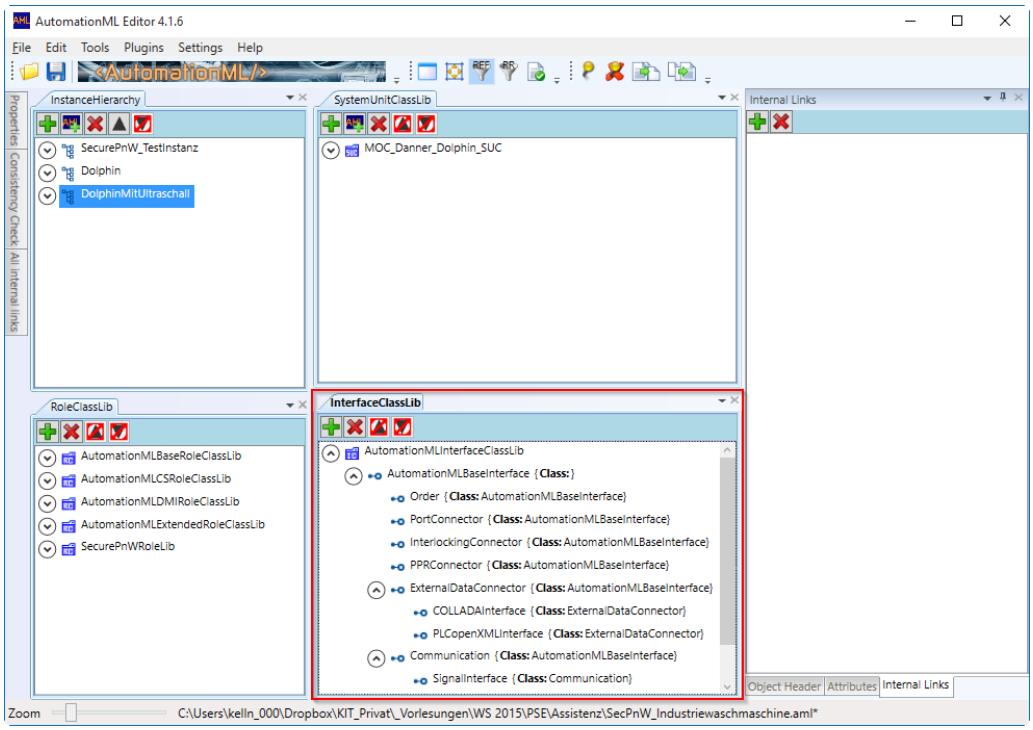


(Abbildung 6: AML Editor mit geöffneter SystemUnitClassLib)
In der SystemUnitClassLib befinden sich schon vorher angefertigte Module. Hier zum Beispiel das Waschmodul oder das Spülmodul.



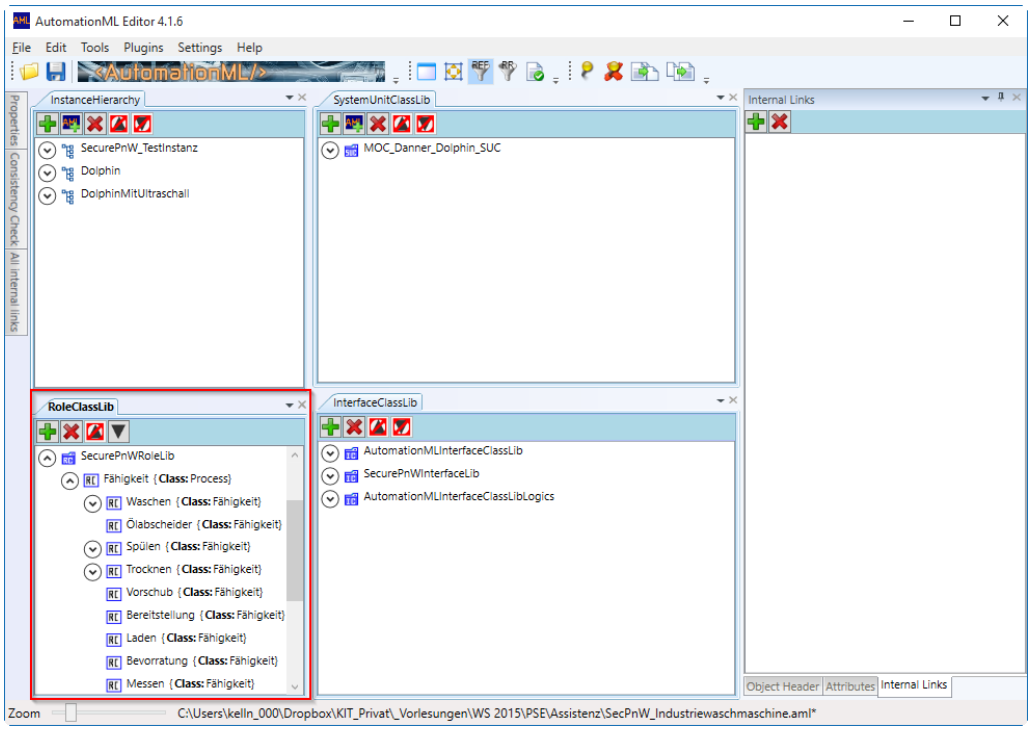
(Abbildung 7: Aufgeklapptes Internal Element im AML Editor)

Öffnet man eines dieser Module, so sieht man die jeweiligen Interfaces, RoleClasses und Attribute. Das Waschmodul hat zum Beispiel die RoleClasses “Ölabscheider” oder “Waschen”.

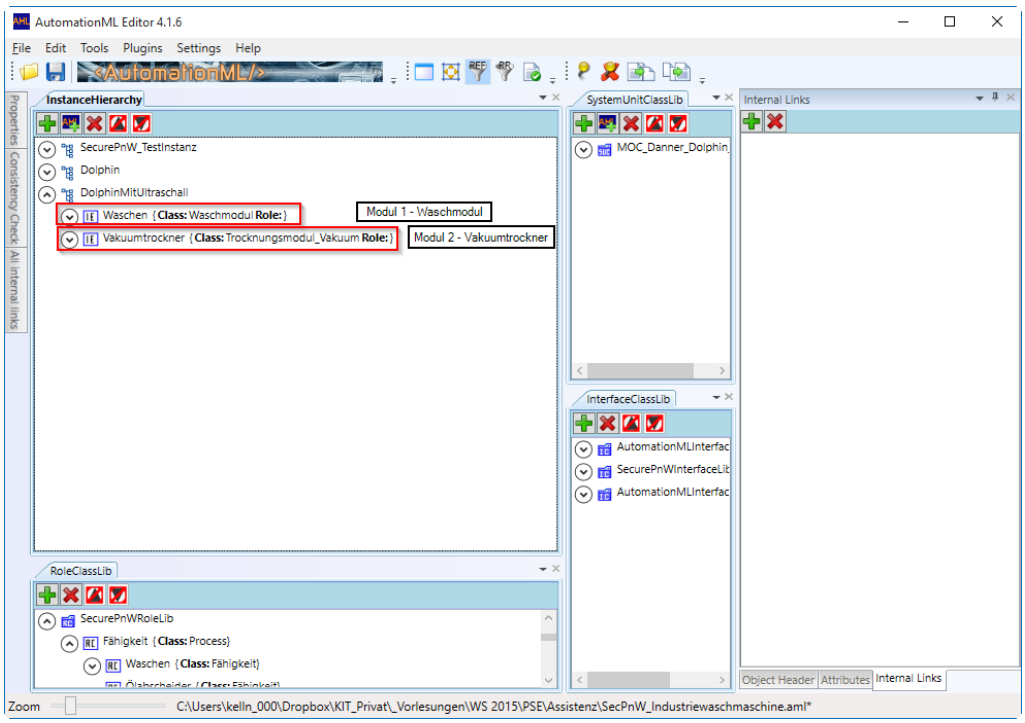


(Abbildung 8: Geöffnete InterfaceClassLib im AML Editor)

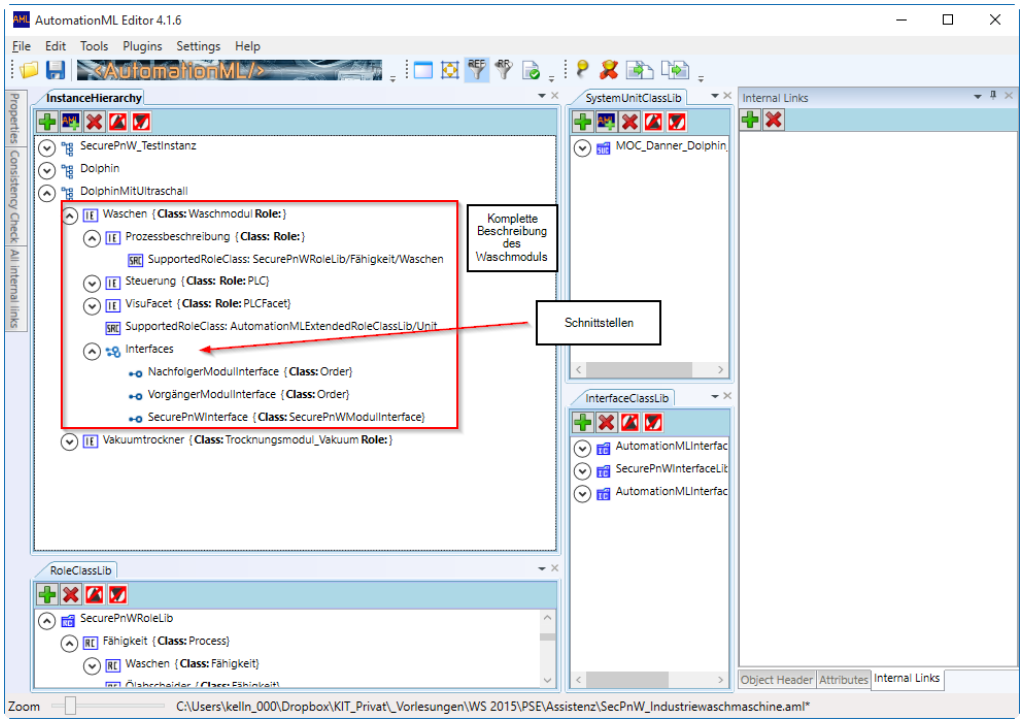
In der InterfaceClassLib befinden sich alle Interfaces, die von Modulen verwendet werden können.



(Abbildung 9: Geöffnete RoleClassLib im AML Editor)
In der RoleClassLib befinden sich Rollen, die Internen Elementen via Drag-and-Drop zugewiesen werden können.

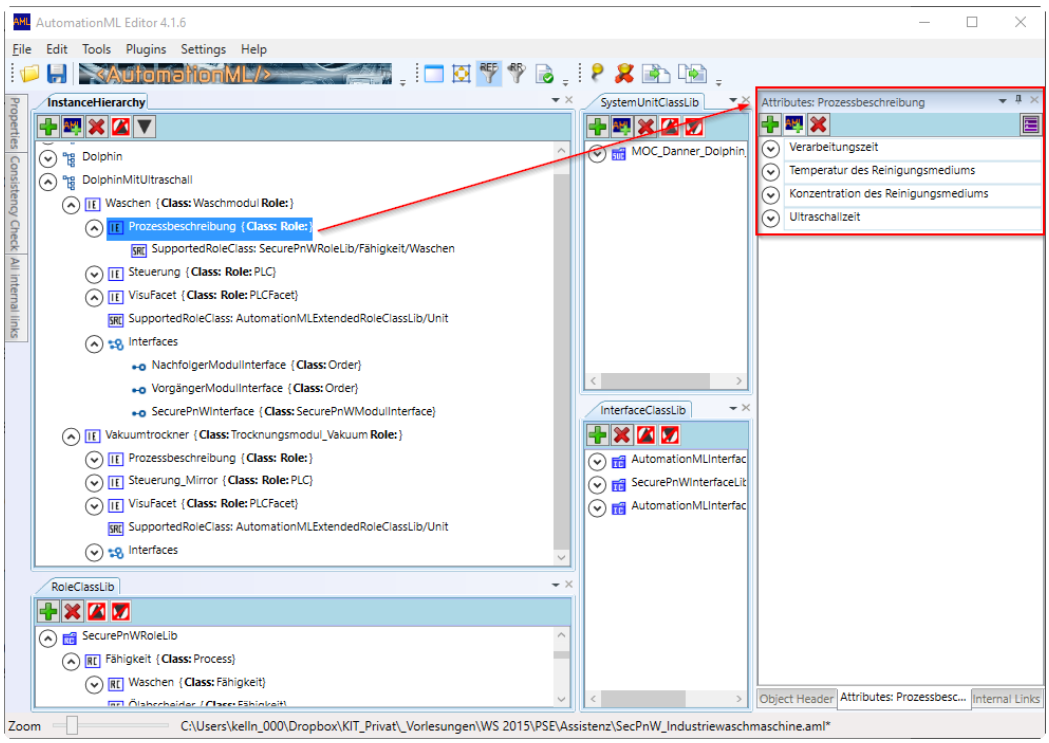


(Abbildung 10: Aufklappen einer InstanceHierarchie im AML Editor)
In der Hierarchie befinden sich unterhalb einer Instanz die Module, die durch InternalElements dargestellt werden. Hier gibt es 2 Module, das Waschmodul und den Vakuumtrockner.



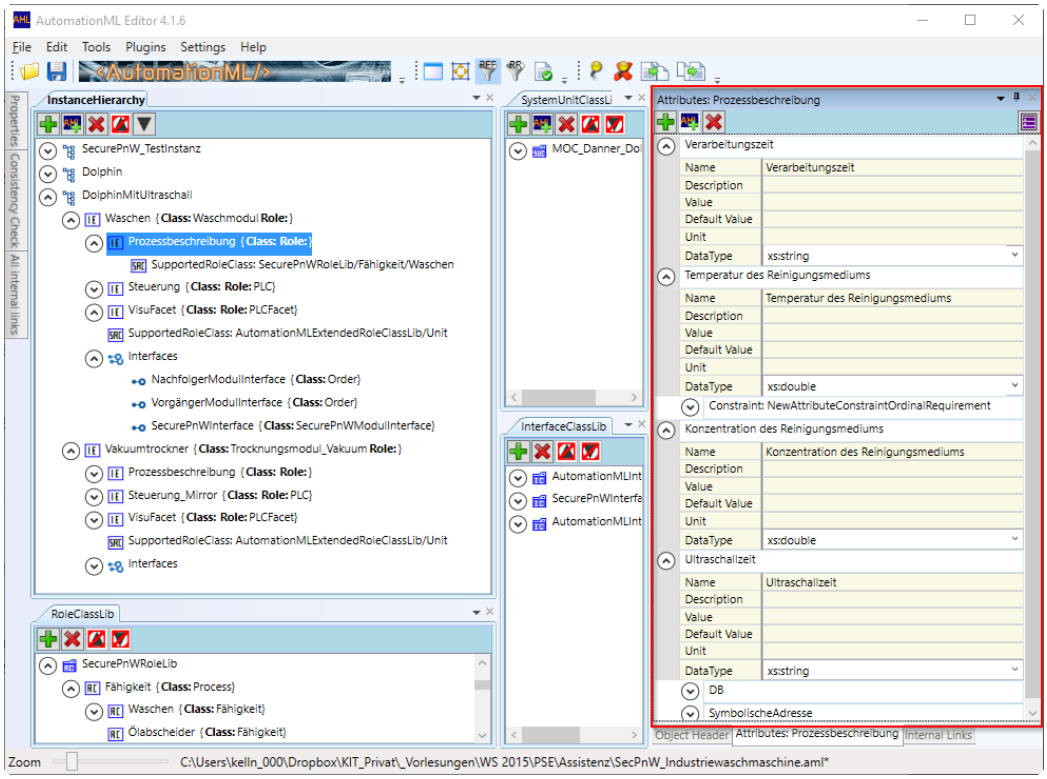
(Abbildung 11: Öffnen eines Internal Elements im AML Editor)

Innerhalb eines Moduls befinden sich wiederum weitere InternalElements, die das Waschmodul beschreiben. Zusätzlich dazu gibt es noch Interfaces.

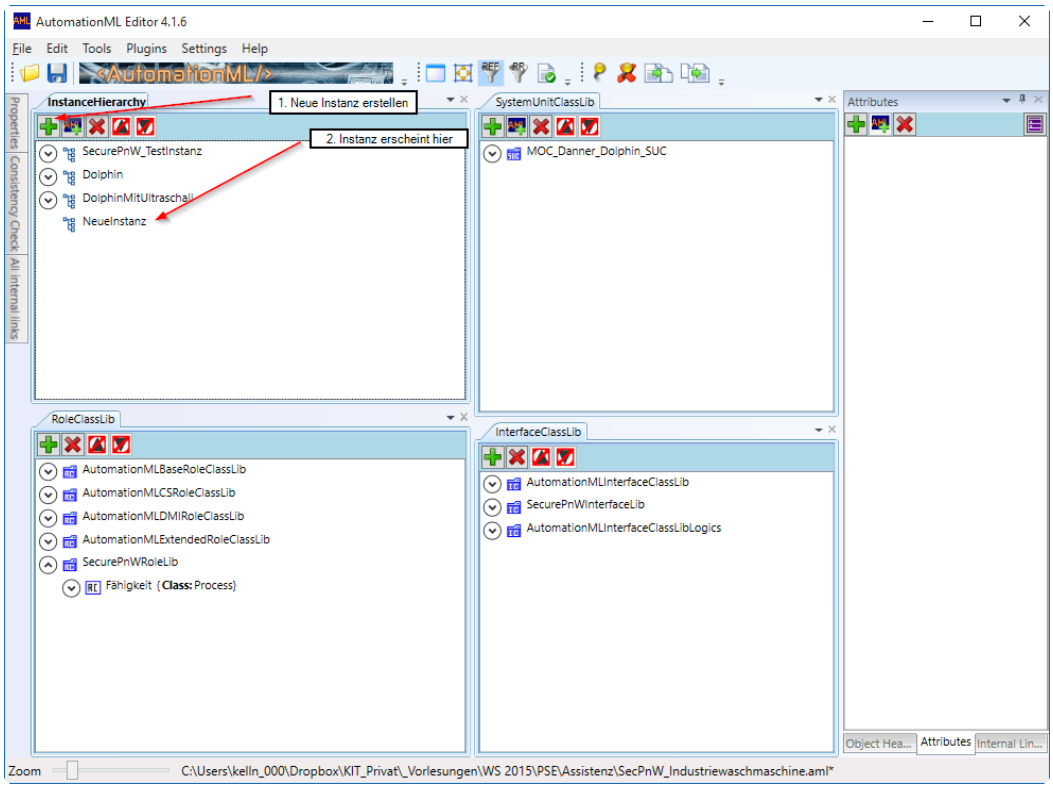


(Abbildung 12: Anzeigen der Attribute eines InternalElements im AML Editor)

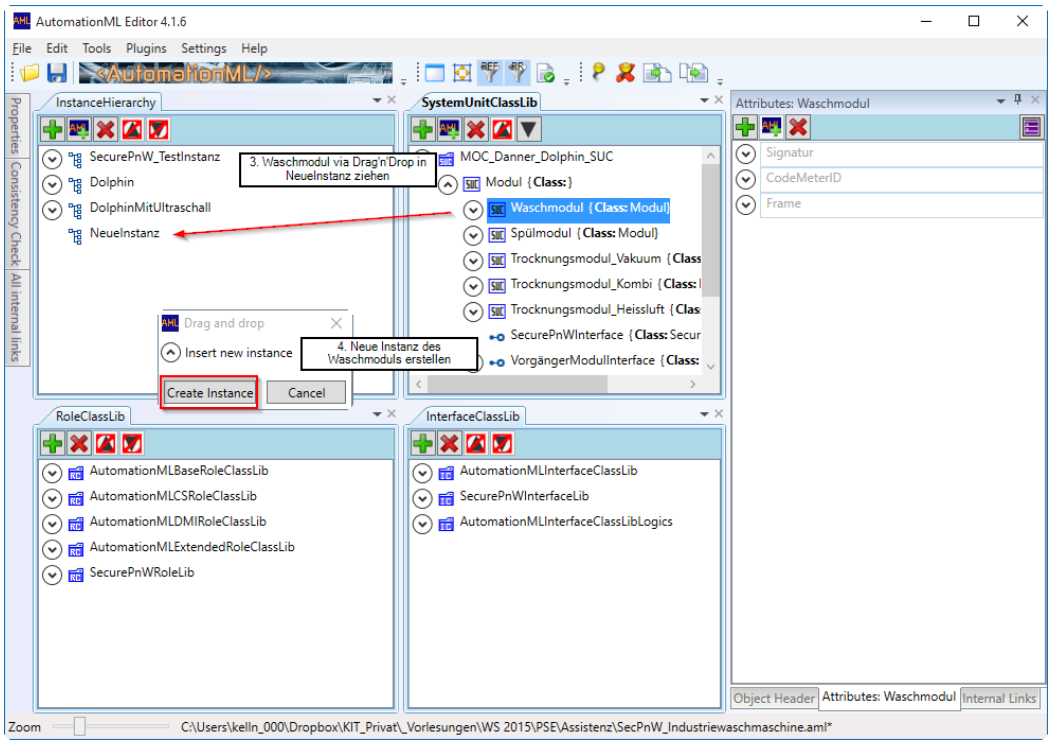
Jedes InternalElement und jedes Interface kann Attribute besitzen, die auf der rechten Seite angezeigt werden.



(Abbildung 13: Detaillierte Aufschlüsselung der Attribute im AML Editor)
Jedes Attribut hat 6 Unterattribute (Name, Description, Value, DefaultValue, Unit und DataType). Attribute können auch geschachtelt sein.

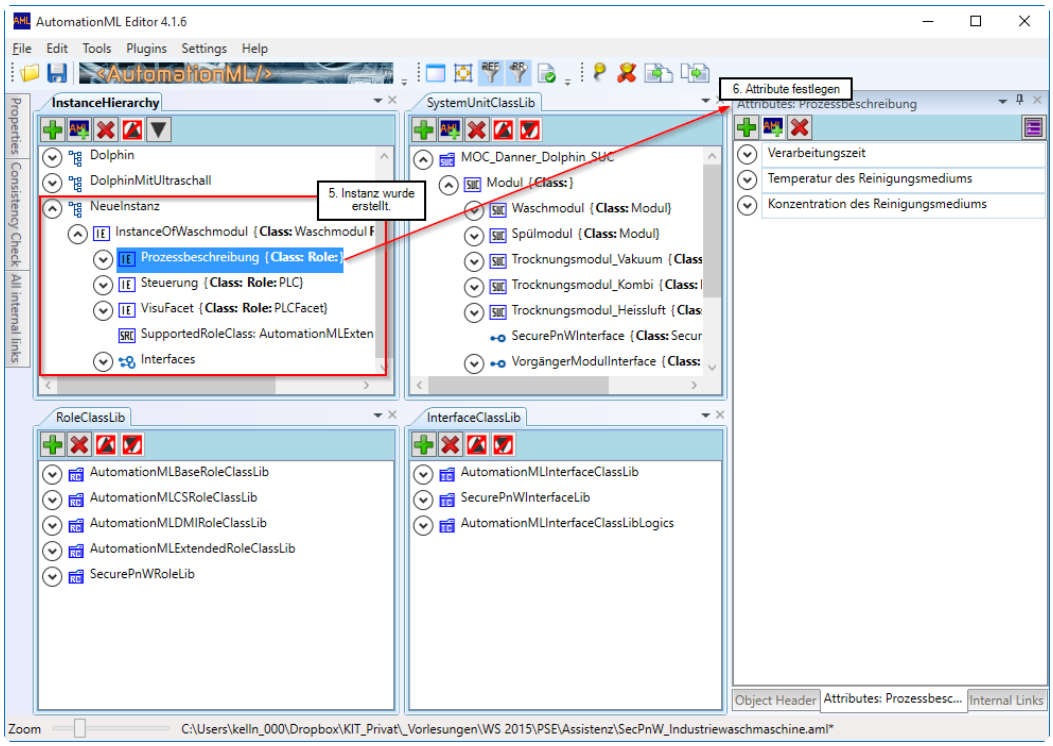


(Abbildung 14: Beschreibung des Prozesses zur Erstellung einer neuen Instanz Hierarchie)
Will man ein neue InstanceHierarchy erstellen, so klickt man auf das grüne Kreuz oben links. Eine neue InstanceHierarchy erscheint daraufhin in der Übersicht.



(Abbildung 15: Hinzufügen eines neuen Elements zu der in Abb. 14 erstellten Instanzhierarchie)

Durch “Ziehen und Loslassen” lässt sich jedes beliebige Modul (hier: Waschmodul) zu der neuen InstanceHierarchy hinzufügen. Durch Klicken auf “Create Instance” wird eine Instanz des Waschmoduls zur neuen InstanceHierarchy hinzugefügt.



(Abbildung 16: Ansicht nach hinzufügen des Waschmoduls)
Die neue Instanz (InstanceOfWaschmodul) wurde erstellt. Durch Klicken auf die InternalElements des Waschmoduls kann man die Attribute nun im rechten Fenster festlegen.

10. Glossar

AML

Siehe AutomationML

Attribut

Einem Element oder anderem Attribut zugeordnete Eigenschaft, die einen Namen, eine Beschreibung, einen Wert, einen Standardwert, eine Einheit und einen Datentyp kapselt.

Attributsnamenzuordnung

Die Abbildung eines Attributs eines Interfaces auf das eines konkreten Elements im zuordnenden Objekt

AutomationML

Rahmenwerk innerhalb von XML, um Produktionsanlagen zu beschreiben. Dient auch dem Austausch von Daten zwischen verschiedenen Programmen.

AutomationML Editor

Editor zur Visualisierung und Editierung von AML-Konzepten und CAEX-Dateien, entwickelt von AutomationML e. V.

CAEX

Siehe Computer Aided Engineering Exchange

Computer Aided Engineering Exchange

Neutrales Datenformat zur Speicherung hierarchischer Objektinformationen, basierend auf XML.

Delta

Ein Delta speichert die Differenz zwischen zwei verschiedenen Ausprägungen eines Elements und ermöglicht es, Funktionen zum Rückgängigmachen von Änderungen zu implementieren.

Extensible Markup Language

Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien. Hier genutzt zur Speicherung der AML-Hierarchien.

Element

Überbegriff für Objekte, die in die Instanz-Hierarchie eingefügt werden können. Elemente können Superelemente haben, die ihnen übergeordnet sind, und Subelemente, die ihnen untergeordnet sind.

Externe Schnittstelle

(engl. “ExternalInterface”) Schnittstelle, die einem Element zugeordnet werden kann und diesem seine Eigenschaften vererbt.

Instanzhierarchie

(engl. “InstanceHierarchy”) Die Gesamtheit aller Elemente in einer AML-Datei, hierarchisch geordnet. Alle Element sind hierbei entweder einem anderen Element untergeordnet oder einem Objekt, welches die Instanzhierarchie selbst repräsentiert und die höchste Ebene darstellt.

Interfacenamenzuordnung

Die Abbildung eines Interfaces auf ein konkrete Element im zuordnenden Objekt.

Internes Element

(engl. “InternalElement”) Ein Standardelement in einer Instanzhierarchie, dem im Gegensatz zu beispielsweise dem Zuordnenden Objekt keine spezielle Funktionalität zugeordnet wird.

Interner Link

(engl. “InternalLink”) Ermöglicht es, Objekte über Hierarchiegrenzen hinweg in Beziehung zu setzen, um beispielsweise eine zeitliche Reihenfolge angeben zu können.

Plugin

Ein Plugin, dt. Einschubprogramm, ist eine Erweiterung eines bestehenden Programms, das für dieses entwickelt und mit diesem zusammen verwendet wird.

Registerkarte

(engl. “Tab”) Darstellungsinstanz, in der ein bestimmtes Element geöffnet ist und detaillierte Informationen über das geöffnete Element angezeigt werden. Mehrere Registerkarten können gleichzeitig existieren, wobei jede ein anderes Element kapselt, jedoch kann zum gleichen Zeitpunkt immer nur eine Registerkarte angezeigt werden.

Rollenklassenbibliothek

(engl. “RoleClassLib”) Externe Bibliothek, die verschiedene importierbare Rollenklassen-Beziehungen enthält.

Schnittstellenbibliothek

(engl. “InterfaceClassLib”) Externe Bibliothek mit Interfaces, die zur Nutzung der Interfaces importiert werden kann

Systemeinheitsklassenbibliothek

(engl. “SystemUnitClassLib”) Externe Bibliothek, die Protoypen für interne Elemente bereitstellt. Diese können, falls die Bibliothek importiert ist, direkt in die Instanzhierarchie hineinkopiert werden

Unterstütze Rollenklasse

(engl. “SupportedRoleClass”) Interne Elemente können verschiedene Rollen innehaben.

XML

Siehe Extensible Markup Language

Zuordnendes Objekt

(engl. “mapping objects”) Ermöglicht es, die Attribute eines Interfaces auf die Attribute eines konkreten Elementes, welches das Interface realisiert, abzubilden.

11. Quellennachweis

Textquellen

[1] Miriam Schleipen, Arndt Lüder, Olaf Sauer, Holger Flatt, Jürgen Jasperneite: Requirements and concept for Plug-and-Work - Flexibility in the context of Industry 4.0 (Anforderungen und Konzept for Plug-and-Work – Flexibilität im Kontext von Industrie 4.0). at - Automatisierungstechnik. Band 63, Heft 10, Seiten 801–820, ISSN (Online) 2196-677X, ISSN (Print) 0178-2312, DOI: 10.1515/auto-2015-0015, October 2015

[2] Wikipedia Industrie 4.0 (Abruf 27.11.2015);

[3] automationml.org und Unterseiten (Abruf 27.11.2015)

Bildquellen

Abruf jeweils am 29.11.2015

KIT-Logo:

http://www.kit.edu/img/intern/kit_logo.png

Fraunhofer-Logo:

http://www.iosb.fraunhofer.de/servlet/is/2670/iosb.gif?command=downloadContent&filename=iosb.gif