Titel: **Individuelle Projekt Arbeit**

Thema: **Matlab/Simulink: Importfunktion für Simulationspaket IMSES in der Gebäudeautomation**

Key Words: IPA, Matlab, Simulink, IMSES, Import

Speicherort:	Local
Dokument Kategorie:	ProjectRecord
Revision:	1.0
Änderungsdatum:	2014-04-25
Dokument Status:	In Bearbeitung
Autor:	Simon Marty
Abteilung:	IC BT CPS R&D ZG CS SAP
Verantwortliche Stelle:	simon.marty@siemens.com
Firma:	Siemens Schweiz AG, Infrastructure & Cities Sector, Building Technologies Division Control Products & Systems
Basierend auf Vorlage:	Workbook_Klein; 3; 2011-09-30; Donat Hutter, 3531

Änderungsgeschichte

Rev	Datum	Autor	Änderungen
0.1	10-Apr-2014	Simon Marty	Status = Neuerstellung
0.2	11-Apr-2014	Simon Marty	Status = **In Bearbeitung**
0.3	14-Apr-2014	Simon Marty	Status = **In Bearbeitung**
0.4	15-Apr-2014	Simon Marty	Status = **In Bearbeitung**
0.5	16-Apr-2014	Simon Marty	Status = **In Bearbeitung**
0.6	17-Apr-2014	Simon Marty	Status = **In Bearbeitung**
0.7	22-Apr-2014	Simon Marty	Status = **In Bearbeitung**
0.8	23-Apr-2014	Simon Marty	Status = **In Bearbeitung**
0.9	24-Apr-2014	Simon Marty	Status = **In Bearbeitung**
1	25-Apr-2014	Simon Marty	Status = **Fertiggestellt**

Inhaltsverzeichnis:

1. Einführung 4
1.1 Zweck des Dokumentes 4
1.2 Zielpublikum 4

2. Projektauftrag 5
2.1 Ausgangslage 5
2.2 Detaillierte Aufgabenstellung 6
2.3 Vorarbeiten, Vorkenntisse 7
2.4 Projektziele 7
2.5 Projektumfang 7

3. Organisatorisches 8
3.1 Datensicherung 8
3.2 Beteiligte Dienste und Fachabteilungen 8
3.3 Verwendete Projektmanagementmethode 8
3.4 Ordnerstruktur 9
3.5 Arbeitsplatz 9
3.6 Risikobeschreibung 9
3.7 Glossary 10
3.8 Quellen 10

4. Zeitplan 11

5. Arbeitsjournal 12
5.1 Zweck des Arbeitsjournals 12
5.2 Anwendungsbereich, Abgrenzung 12
5.3 Aufbau 12
5.4 Arbeitsjournale vom 10.04.2014 bis 25.04.2014 13

6. Management Summary 23
6.1 Ausgangslage 23
6.2 Umsetzung 23
6.3 Erwartetes Ergebnis 23

7. Analyse 24
7.1 Ziel und Zweck der Erweiterung 24
7.2 Ist Zustand 25
7.2.1 Bestehende Umgebung 25
7.3 Export 26
7.3.2 Tool Analyse 31
7.3.3 Sequenzdiagramm 35
7.4 Soll Zustand 35

8. Planung 36
8.1 Fehlersituationen 36
8.1.1 Fall 1 36
8.1.2 Fall 2 37
8.1.3 Fall 3 42

9. Implementierung & Umsetzung 43
9.1 Struktogramme 43
9.2 getXmlChartName 45
9.3 getUniqueObjID 45
9.4 XML Chart Datei 46
9.5 Implementierung Regelwerk 46
9.6 Erzeugung eines Objektes 46

10. Testing 47
10.1 Testumgebung 47
10.2 Testfälle 47
10.2.1 Funktionalitätstests 48
10.2.2 White Box-Tests 49
10.2.3 Testsummary 53
11. Schlusswort 56
12. Anhang 57
12.1 BA_OBJ.m 57

Abbildungsverzeichnis:

Abbildung 3–1: IPERKA 8
Abbildung 3–2: Ordnerstruktur 9
Abbildung 7–1: IPA - Umgebung 25
Abbildung 7–2: Einsatzbereich IMSES 26
Abbildung 7–3: Aufbau Chart: XML Ansicht 27
Abbildung 7–4: Aufbau Chart: Simulink Ansicht 27
Abbildung 7–5: Aufbau Chart: Aufbau Simulink 28
Abbildung 7–6: .ba Datei Tree-View 29
Abbildung 7–7: .ba Datei Tree-View 30
Abbildung 7–8: .ba Datei Tree-View 30
Abbildung 7–9: Import Ablauf - IMSES 31
Abbildung 7–10: Funktionsteilung - IMSES 31
Abbildung 7–11: SubList Beispiel 31
Abbildung 7–12: Objekt Klassen 32
Abbildung 7–13: ObjectList - Beispiel 33
Abbildung 7–14: Objekt Hierarchie 33
Abbildung 7–15: Ablauf Diagramm - IMSES 34
Abbildung 7–16: Sequenzdiagramm 35
Abbildung 8–1: Objekt ID - Tree-View 36
Abbildung 8–2: Activity Diagramm - Fall 1 36
Abbildung 8–3: Fall 2 - Tree View 37
Abbildung 8–4: Ausschnitt XML Chart File 38
Abbildung 8–5: Ausschnitt des Regelwerks 39
Abbildung 8–6: Struktogramm: Regelwerk 40
Abbildung 8–7: Activity Diagramm - Fall 2 41
Abbildung 8–8: Activity Diagramm - Fall 3 42
Abbildung 9–1: Struktogramm Fall 1 43
Abbildung 9–2: Struktogramm Fall 2 & 3 44
Abbildung 10–1: msinfo32 System Summary 47

Tabellenverzeichnis:

Tabelle 2–1: Projektauftrag 5
Tabelle 2–2: Fälle 1-3 6
Tabelle 3–1: Dienste & Fachabteilungen 8
Tabelle 3–2: Risikobeschreibung 9
Tabelle 3–3: Quellenverzeichnis 10
Tabelle 7–1: XML Aufbau Beschreibung 28
Tabelle 8–1: Regelwerk in Array - Beispiel 40
Tabelle 8–2: Regelwerk integrieren Evaluation 41
Tabelle 9–1: Funktionsbeschreibung 45
Tabelle 9–2: Funktionsbeschreibung 45

1. Einführung

1.1 Zweck des Dokumentes

Dieses Dokument enthält die Anforderungen, das Design und die Umsetzung der Importfunktion für Simulationspaket IMSES in der Gebäudeautomation, welche im Rahmen der IPA von Simon Marty implementiert wird.

1.2 Zielpublikum

Dieses Dokument richtet sich in erster Linie an die Experten und Betreuer der IPA. Dadurch kann die Arbeit nachvollzogen und beurteilt werden.
Im Nachhinein kann dieses Dokument von den nachfolgenden Entwicklern von IMSES verwendet werden. Sie können sich eine Übersicht über die Erweiterung des Imports verschaffen, um diese zu verändern, verbessern oder bearbeiten.

2. Projektauftrag

Tabelle 2–1: Projektauftrag

Projekttitel	Importfunktion für Simulationspaket IMSES in der Gebäudeautomation
Prüfungskandidat	Simon Marty
Fachvorgesetzter	Michael Speckien
Auftraggeber	Michael Speckien

2.1 Ausgangslage

Für die Gebäudeautomation mit frei programmierbaren Controllern wird bei SIEMENS ein graphisches Engineering Tool (CFC) eingesetzt, mit welchem man Automationsaufgaben graphisch programmiert.
Im CFC werden Bausteine auf einem Plan platziert untereinander verschaltet. Die so entstandene Automationssoftware wird unter anderem mit Matlab / Simulink (R) getestet.
Da Simulink-Pläne ähnlich aufgebaut sind wie CFC-Pläne, ist es möglich, diese in Simulink nachzubauen und dort zu testen.
Die CFC-Pläne können exportiert werden (ABT-Export). Mit einem Tool (IMSES) können die exportierten Daten anschliessend nach Simulink importiert werden und dort getestet werden.

Die Automationssoftware besteht aus einzelnen Räumen (Area). Jeder Raum besteht aus einzelnen Modulen (Applikationsfunktionen). Als Schnittstelle zwischen den Räumen, den Applikationsfunktionen, den physikalischen Ein-/Ausgängen und dem Endanwender dienen BA-Objekte. BA Objekte sind hierarchisch strukturiert. Die Struktur wird über Parent-Child-Verweise dargestellt, wobei ein Child-Objekt zu mehreren Parent-Objekten gehören kann.

BA-Objekte gehören jeweils zu einem Raum oder einer Applikationsfunktion (Owned). Applikationsfunktionen können auf ihre "owned" Objekte zugreifen. Alle BA-Objekte können aber auch von anderen Applikationsfunktionen gelesen oder beschrieben werden (Connected). Beim Testen einzelner Räume oder Applikationsfunktionen fehlen die "connected" Objekte, so dass ein Test nur unvollständig möglich ist. Um einen vollständigen Test durchzuführen, müssen die fehlenden Objekte derzeit vom Anwender im Simulink manuell erstellt werden.

Der ABT-Export besteht aus einer .ZIP-Datei. Extrahiert ergeben sich:

- Eine .ba-Datei im XML-Format mit einer Tabelle aller Objekte mit ihren Properties einschliesslich der Hierarchiestruktur und der owned/connected-Information
- Eine .XML-Datei mit allen im CFC Plan vorhandenen Bausteinen sowie ihren Verschaltungen und Parametrierungen

2.2 Detaillierte Aufgabenstellung

Um die aufwändige, manuelle und somit fehlerbehaftete Eingabe der fehlenden BA-Objekte zu vermeiden, sind folgende Erweiterungen des Simulink-Importmoduls vorgesehen:

Tabelle 2–2: Fälle 1-3

Fall 1	Ein Objekt in der BA Datei hat einen Verweis auf ein Child-Objekt (Subordinate_List). Das Child-Objekt ist nicht im BA Datei vorhanden. Ursache: Inkonsistenz im zu testenden Modul. Lösung: Das Objekt muss erstellt werden. Der Verweis beinhaltet die notwendigen Informationen. Fehlermeldung an Anwender.
Fall 2	Ein Objekt in der BA-Datei hat einen Platzhalter (Subordinate_Annotation) für einen Verweis auf ein Child-Objekt. Es ist kein Verweis (in Subordinate_List) vorhanden, aber das Objekt wird im CFC Chart verwendet (X-FB). Ursache: a) „Connected": Objekt ist nicht im zu testenden Modul enthalten sondern in einem anderen. b) „Owned“: Inkonsistenz im zu testenden Modul Lösung: Das Objekt muss erstellt werden. Der Baustein im CFC-Chart enthält die benötigten Informationen. Ist das erstellte Objekt nicht eindeutig definiert oder handelt es sich um ein „Owned“ Objekt, erfolgt eine Fehlermeldung an den Anwender.
Fall 3	Ein Objekt in der BA-Datei hat einen Platzhalter für einen Verweis auf ein Child-Objekt. Es ist kein Verweis vorhanden, und das Objekt wird nicht im CFC Chart verwendet. Ursache: Das Objekt hat keine Funktion, sondern ist reine Information. Lösung: Kein Objekt erstellen. Bei „owned“ erfolgt eine Fehlermeldung an den Anwender

Ein Child-Objekt kann mehrere Parent-Objekte haben. Daher dürfen Objekte nur dann neu angelegt werden, wenn sie nicht bereits vorher als Child eines anderen Parent-Objekts angelegt worden sind.

„Owned“ Objekte werden in der importierten Area angelegt, für alle „Connected“ Objekte zusammen wird eine neue Area erstellt.

2.3 Vorarbeiten, Vorkenntisse

Das Import Tool IMSES wurde bereits vor der IPA mit Export Daten zur Verfügung gestellt. Der Ablauf des Programms wurde grob analysiert, um eine Übersicht zu schaffen. Ausserdem wurde eine Prozedur erstellt, die ein XML Datei durchsuchen kann, um die MATLAB Syntax und die MATLAB Entwicklungsumgebung kennenzulernen.
Weitere Vorarbeiten wurden nicht erledigt.

2.4 Projektziele

Im Rahmen der IPA soll der vorhandene Simulink Import so angepasst werden, dass alle Fälle vollständig abgedeckt sind.

2.5 Projektumfang

Folgende Tätigkeiten sind im Rahmen der IPA durchzuführen:

- Analyse der bestehenden Importfunktion
- Design der Erweiterungen wie oben beschrieben
- Implementierung innerhalb der bestehenden Importfunktion
- Test der Implementierung: White-Box-Ansatz mit Stichproben
- Erstellen der Projektdokumentation und der technischen Dokumentation

3. Organisatorisches

3.1 Datensicherung

Die gesamte Ordnerstruktur inklusive dem Code befindet sich auf einem Siemens Server, von welchem täglich ein Backup erstellt wird. Zudem wird von der ganzen Arbeit täglich eine ZIP Datei erstellt, welche auf einer lokalen Festplatte abgelegt wird.

3.2 Beteiligte Dienste und Fachabteilungen

Tabelle 3–1: Dienste & Fachabteilungen

Firma	Siemens Schweiz AG, Building Technologies Division
Abteilung	IC BT CPS R&D ZG CS SAP
Verwendete Software	- MATLAB R2011b - IMSES - Microsoft Office 2007 - Microsoft Visio - Notepad++ - HUS Struktogrammer - XML Notepad 2007
Verwendete Tools	Snipping Tool Msinfo32.exe

3.3 Verwendete Projektmanagementmethode

Bei der Arbeit an dem Projekt arbeite ich nach dem Schema von IPERKA:

1.) **I**nformationen beschaffen
2.) **P**lanen
3.) **E**ntscheiden
4.) **R**ealisieren
5.) **K**ontrollieren
6.) **A**uswerten

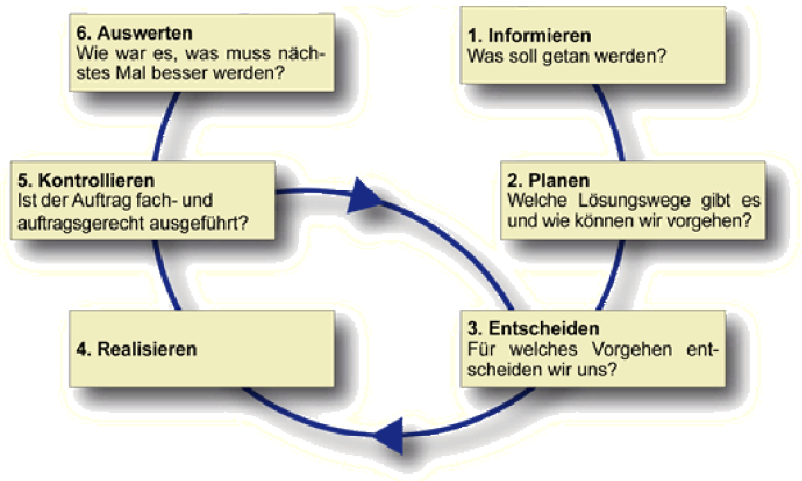


Abbildung 3–1: IPERKA

3.4 Ordnerstruktur

Für die Durführung der IPA werden alle relevanten Daten zentral an einem Ort abgelegt. Für die Ablage wurde die folgende Struktur verwendet:

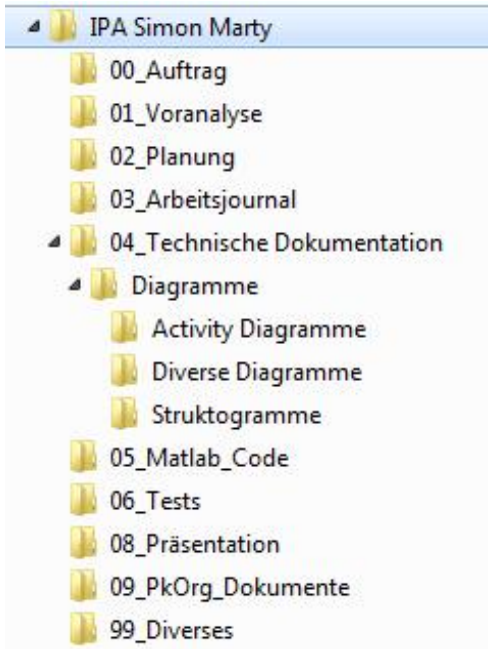


Abbildung 3–2: Ordnerstruktur

3.5 Arbeitsplatz

Der Arbeitsplatz befindet sich in Zug am Zählerweg 7 im vierten Stock. Zur Verfügung steht ein Fujitsu Laptop mit zusätzlichem Monitor.

3.6 Risikobeschreibung

Hier wird beschrieben was den Ablauf der IPA gefährden könnte. Zum Bsp. technische Machbarkeit oder Abhängigkeit von Dritten.

Tabelle 3–2: Risikobeschreibung

Zeitplan	Im Zeitplan wurde eine gewisse Reservezeit eingeplant. Diese befindet sich jedoch erst am Schluss und vor der Abgabe der Arbeit. Somit können allfällige Verzögerungen erst am Ende wieder korrigiert werden. Bei einer Abweichung vom Zeitplan zum Start der IPA würde sich dementsprechend die ganze Planung verschieben. Besser wären mehrere Blöcke an Reservezeit, aufgeteilt auf die gesamte Zeitspanne, gewesen.
Matlab Kenntnisse	Da nicht auf sehr viel Routine im Umgang mit Matlab zurückgegriffen werden kann, könnten sich während der Implementierung unvorhergesehene Verzögerungen ergeben.

3.7 Glossary

Name	Beschreibung
.ba Datei	Tabelle aller Objekte mit ihren Properties einschliesslich der Hierarchiestruktur und der owned/connected-Information (XML-Format)
ABT	Automation Building Tool Tool zum Engineering der Gebäudeautomation, basierend auf dem TIA Portal. Beinhaltet unter anderem den CFC.
Applikationsfunktion	Einzelnes Modul der Automationssoftware einer Area (z.B. Beschattung)
Area	Objekt im Simulink das einen Raum oder ein Teil eines Raumes repräsentiert
BA Objekt	Building Automation Objekt. Bildet die Schnittstelle zwischen physikalischen Ein- und Ausgangssignalen und dem Chart, sowie die Schnittstelle zum Bedienen und Beobachten für den Endkunden. Beispiel: analoger Eingang für Raumtemperatur.
CFC	Continuous Function Chart Editor. Graphische Programmiersprache zur Programmierung von Automationslösungen.
CFC-Chart	Plan, auf dem der Engineerer mit dem CFC-Editor Bausteine platzieren, parametrieren und untereinander verschalten kann. Die Regelung- und Steuerungslogik für die Automation der HLK-Geräte, Licht und Jalousien in einem Raum wird grafisch mit Charts und Bausteinen programmiert.
Connected	Greift eine Applikationsfunktion auf ein BA-Objekt einer Area oder einer anderen Applikationsfunktion zu, ist das BA Objekt zur Applikationsfunktion „connected“.
IMSES	**I**nterface **M**atlab/**S**imulink **E**ngineering **S**ystem. Interface zwischen der auf Matlab/Simulink basierten Entwicklungs- und Testumgebung zum Engineering System des Desigo Systems.
MATLAB	Software der Fa. Mathworks Inc zur numerischen Lösung mathematischer Probleme.
Owned	Jedes BA Objekt wird entweder einer Area oder einer Applikationsfunktion zugeordnet. Es ist „owned“ von der Area oder Applikationsfunktion.
SIMULINK	Zusatzprodukt (Toolbox) zu MATLAB zur graphischen Programmierung mit Hilfe von Bausteinen. Bei SIEMENS verwendet für die Programmierung von HLK-Streckenmodellen und den Reglerentwurf.
TIA Portal	Totally Integrated Automation Portal Toolset zum Engineering von Automationslösungen
White Box Test	Test bei dem man Zugriff auf den Code hat
ZIP	Komprimiertes Dateiformat.

3.8 Quellen

Tabelle 3–3: Quellenverzeichnis

Thema	Quelle	Datum
Projektauftrag	https://extranet.pkorg.ch/	10.04.2014
IPERKA	http://tgabathuler.ch/IPERKA/Index.html	10.04.2014
Aktivitätsdiagramm	http://de.wikipedia.org/wiki/Aktivit%C3%A4tsdiagramm	11.04.2014
White Box Test	http://de.wikipedia.org/wiki/White-Box-Test	17.04.2014
Matlab Hilfe	http://www.mathworks.ch/matlabcentral/	14.04.2014

4. Zeitplan

	IPA Simon Marty	Abhängigkeit			Aufwand			Status		Geplanter Ablauf											
Arbeitsschritt	Tätigkeiten	Voraussetzung	Priorität	Nächster Schritt	Soll [h]	Ist [h]	Abweichung [%]	Meilenstein [Datum]	Status	10.04.2014	11.04.2014	14.04.2014	15.01.1900	16.04.2014	17.04.2014	18.04.2014	21.04.2014	22.04.2014	23.04.2014	24.04.2014	25.04.2014
#100	**Projektmanagement**																				
#101	Zeitplanung "IST" nachtragen		1		1.0	1.0	0			0.1	0.1	0.1	0.1	0.1	0.1			0.1	0.1	0.1	0.1
#102	Arbeitsjournal führen (0.5h pro Tag)		1		5.0	5.0	0			0.5	0.5	0.5	0.5	0.5	0.5			0.5	0.5	0.5	0.5
#103	Technische Dokumentation führen		1		26.0	27.3	-5			4.0	2.9	0.9	0.5	1.0	5.0			7.0	6.0		
#200	**Vorbereitung**																				
#201	Tätigkeiten und Meilensteine erfassen		1	#202	2.0	2.0	0			2.0											
#202	Zeitplanung "SOLL" erstellen	#201	1		1.0	1.0	0			1.0											
#203	Entwicklungsumgebung Einrichten		1		1.0	0.2	80			0.2											
#204	Dokumentenvorlage erstellen		1		1.0	0.2	80			0.2											
#300	**Analyse und Design**																				
#301	Vorhandener Code analysieren und Schlüsselstellen finden		1		3.0	3.5	-16.7				1.5	2.0									
#302	Activity Diagramme erstellen	#301	1	#305	2.0	2.0	0				2.0										
#303	Struktogramme erstellen	#301	1		2.0	3.4	-70				1.0				2.4						
#400	**Implementation und Umsetzung**																				
#401	Findung von nicht existierenden Objekten		1	#402	1.0	1.0	0					1.0									
#402	Findung von fehlenden Einträgen in Subordinate_List		1	#403	3.0	2.0	33.3					2.0									
#403	Findung von Inkonsistenzen in Subordinate_List und Subordinate_Annotations		1	#404	3.0	0.9	70						0.5	0.4							
#404	Erzeugung von Objekten in Simulink	#401	1		6.0	7.9	-31.7					1.5	6.4								
#405	Bearbeitung von Objekten in Simulink		1		4.5	6.0	-33.3							6.0							
#500	**Testing**																				
#501	Testfälle ermitteln		1		2.0	1.5	25													1.5	
#502	Code Testen	#501	1	#503	3.0	2.4	20													2.4	
#503	Testdokumentation führen	#502	1		3.0	3.5	-16.7													3.5	
#600	**Abschluss**																				
#601	Arbeitsjournal abschliessen		1		0.5	0.7	-40														0.7
#602	Zeitplan abschliessen		1		0.5	0.5	0														0.5
#603	Technische Dokumentation abschliessen		1		1.0	4.0	-300														4.0
#700	**Termine / Meilensteine**																				
#701	Gespräch mit Erstexperten		1		1.5	1.0	33.3		OK									1.0			
#702	Analyse und Design abgeschlossen		1																		
#703	Implementation abgeschlossen		1																		
#704	Projektabgabe (Binden, hochladen)		1		1.0	1.0	0	25.04.2014													1.0
#800	**Pufferzeit**																				
#801	Reservezeit				6.0	2.0	66.7												2.0		
	Total:				80.0	80.0	0.0			8.0	8.0	8.0	8.0	8.0	8.0			8.6	8.6	8.0	6.8

Karfreitag

Ostermontag

x	Geplanter Ablauf (x = Anzahl effektiv benötigte Stunden)
	Abweichungen vom geplanten Ablauf
	Meilenstein
	Abgabetermin

5. Arbeitsjournal

5.1 Zweck des Arbeitsjournals

Im Arbeitsjournal werden die täglichen Arbeiten, aufgetretene Probleme sowie allfällige Hilfestellungen festgehalten. Dabei dient das Arbeitsjournal der Orientierung über den Stand des Projektes an den jeweiligen Arbeitstagen.

5.2 Anwendungsbereich, Abgrenzung

Das Arbeitsjournal ersetzt keine Dokumentation und ist nur im Zusammenhang mit der IPA von Interesse. Es zeigt den Fortschritt und die Entwicklung der Arbeit.

5.3 Aufbau

In jedem Journal werden zuerst die für den Tag gesetzten Ziele genannt, wie sie im Zeitplan aufgeführt sind. Danach wird für jedes Ziel das Vorgehen und die erreichten Fortschritte festgehalten. Allfällige Hilfestellungen, relevante Ereignisse und der E-Mail Verkehr sind ebenfalls Teil des Arbeitsjournals und werden entsprechend aufgeführt. Abschliessend wird für jeden Tag ein Fazit gezogen und die geleistete Arbeit hinterfragt.

5.4 Arbeitsjournale vom 10.04.2014 bis 25.04.2014

Montag, 10.04.2014

Gesetzte Ziele	#201 Tätigkeiten und Meilensteine erfassen #202 Zeitplanung „Soll“ erstellen #203 Entwicklungsumgebung einrichten #204 Dokumentenvorlage erstellen
Tätigkeiten und Meilensteine	Als Vorbereitung für den Zeitplan habe ich als Erstes alle Tätigkeiten auf Papier erfasst. Danach habe ich mir die wichtigsten Punkte aus der Liste von Tätigkeiten notiert und diese als Meilensteine festgelegt.
Zeitplanung „Soll“	Basierend auf den ermittelten Tätigkeiten und Meilensteine konnte ich dann mit dem Zeitplan beginnen. Dieser Task hat jedoch ein wenig mehr Zeit in Anspruch genommen, als ich es geplant hatte. Während des Eintragens kamen mir laufend neue Tätigkeiten in den Sinn, welche aus meiner Sicht ebenfalls im Zeitplan aufgeführt werden sollten.
Entwicklungs-Umgebung	Ich habe auf meinem Rechner die Entwicklungsumgebung für die spätere Implementation eingerichtet und alle benötigten Ressourcen zusammengetragen.
Dokumentenvorlage	Für die technische Dokumentation habe ich eine Vorlage, basierend auf einem Siemens Dokument, erstellt. Ich habe mir bereits erste Gedanken über Struktur und Aufbau gemacht und diese in Form eines Inhaltsverzeichnisses festgehalten.
Hilfestellungen	-
Probleme	-
Mailverkehr	-
Sonstiges	Ist-Zeiten wurden eingetragen und ein Backup wurde erstellt.
Fazit	Ungeachtet der Zeit die ich zugunsten des Zeitplanes einbüsste, ist der erste Tag meiner IPA sehr gut verlaufen. Ein Mehraufwand für die detaillierte Erstellung des Zeitplanes war hier aus meiner Sicht absolut sinnvoll und wird mir im weiteren Verlauf der Arbeit sicher zugunsten kommen.

Freitag, 11.04.2014

Gesetzte Ziele	#301 Vorhandener Code analysieren und Schlüsselstellen finden #302 Activity Diagramme erstellen #203 Struktogramme erstellen
Code Analyse	Damit ich mir einen Überblick verschaffen kann, in welchem Umfang Veränderungen vorgenommen werden müssen, und vor allem wie und wo sie vorgenommen müssen, analysierte ich den vorhandenen Code. Es ging länger als erwartet. Jedoch ist so eine Tätigkeit sehr schwierig zum abschätzen, da man nicht wissen kann, was auf einen zukommt
Activity Diagramme	Um die Problemlösung der Fälle 1,2 & 3 übersichtlich darzustellen, habe ich pro Fall ein Activity Diagramm erstellt.
Struktogramme	Um den Ablauf genau zu planen und darzustellen, habe ich Struktogramme erstellt. Diese werden mir später bei der Implementierung helfen, sofern sie korrekt sind und nicht geändert werden müssen.
Hilfestellungen	Activity Diagramm: http://de.wikipedia.org/wiki/Aktivit%C3%A4tsdiagramm
Probleme	Am Anfang war ich ein wenig unschlüssig bei der Strukturierung des Planung Teils, was zu einem kleinen Zeitverzug kam.
Mailverkehr	-
Sonstiges	Ist-Zeiten wurden eingetragen und ein Backup wurde erstellt. Zeitplan wurde überarbeitet.
Fazit	Der heutige Tag verlief grundsätzlich gut. Ich hätte gedacht, dass ich mit der Doku ein wenig schneller vorankomme. Der Verzug ist jedoch überhaupt nicht gross und daher auch nicht problematisch. Die Erstellung der Struktogramme war auch zeitintensiver und kniffliger als ich zuerst dachte.

Montag, 14.04.2014

Gesetzte Ziele	#301 Vorhandener Code analysieren und Schlüsselstellen finden #401 Findung von nicht existierenden Objekten #404 Erzeugung von Objekten in Simulink
Code Analyse	Es gab weiterhin Unklarheiten im Code, somit musste ich mich noch einmal dahinter setzen, und relevante Funktionen und Code Stellen analysieren.
Findung von nicht existierenden Objekten	Nach gründlicher Analyse musste ich feststellen, dass diese Funktion bereits existiert, jedoch am falschen Ort. Ich musste sie etwas anpassen und am richtigen Ort einsetzen.
Erzeugung von Objekten in Simulink	Das Erzeugen von neuen Objekten ging ich anfangs falsch an. Später realisierte ich, dass es eine viel einfachere Lösung gab. Somit war die vorherige Arbeit umsonst.
Hilfestellungen	-
Probleme	Beim Auslesen von XML Parameter gab es kleinere Syntax Probleme, welche eine relative grosse Verzögerung bewirkte. Schlussendlich konnte ich das Problem beheben und gemäss Zeitplan fortfahren.
Mailverkehr	-
Sonstiges	Ist-Zeiten wurden eingetragen und ein Backup wurde erstellt.
Fazit	Der Tag verlief gut. Der Start der Implementation gab mir einen Motivationsschub, da es grosse Abwechslung und Spannung in die IPA bringt. Es gab kleinere Verzögerungen, welche jedoch nicht weiter tragisch sind.

Dienstag, 15.04.2014

Gesetzte Ziele	#403 Findung von Inkonsistenzen in Subordinate_List und Subordinate_Annotations #404 Erzeugung von Objekten in Simulink
Findung von Inkonsistenzen	Diese Implementation verlief problemlos und ging schnell von statten.
Erzeugung von Objekten in Simulink	Das Erzeugen von neuen Objekten ging ich anfangs komplett falsch an. Später realisierte ich, dass es eine viel einfachere Lösung gab. Somit war die vorherige Arbeit umsonst.
Hilfestellungen	Fragen bezüglich des XML Chart Dateipfad wurden mit Thomas Weiss geklärt.
Probleme	Der Pfad zur XML Chart Datei konnte nicht übergeben werden. Der Dateiname musste aus dem Grundpfad herausgeschnitten werden. Es gab ausserdem Probleme mit vorhandenem Code, da sich neuer Code mit dem Alten überschnitt.
Mailverkehr	-
Sonstiges	Ist-Zeiten wurden eingetragen und ein Backup wurde erstellt.
Fazit	Heute bin ich mit der Implementation gut vorangekommen, obwohl es kleinere Probleme gab. Ich hatte jedoch die Dokumentation zu wenig beachtet, was ich morgen noch nachholen muss.

Mittwoch, 16.04.2014

Gesetzte Ziele	#403 Findung von Inkonsistenzen in Subordinate_List und Subordinate_Annotations #405 Bearbeitung von Objekten in Simulink
Bearbeitung von Objekten	Ich ging die Sache zuerst von der falschen Seite an. Ich hätte noch mehr Zeit in die Analyse stecken sollen, dann hätte ich das vielleicht verhindern können. Schlussendlich konnte ich die Lösung elegant in den vorhandenen Code integrieren.
Hilfestellungen	-
Probleme	Eine kleine Zeitverzögerung bei der Implementierung von #405 trat auf.
Mailverkehr	-
Sonstiges	Ist-Zeiten wurden eingetragen und ein Backup wurde erstellt.
Fazit	Auch der heutige Tag verlief grundsätzlich gut. Ich kam gut voran beim Programmieren. Es funktioniert jetzt alles und alle Fälle sind abgedeckt. Ich bin froh, die Implementation fertig stellen zu können.

Donnerstag, 17.04.2014

<table>
<tr><td>Gesetzte Ziele</td><td>#103 Technische Dokumentation führen</td></tr>
<tr><td>Dokumentation</td><td>Da ich mit der Implemenation fertig bin, habe ich noch einmal die Struktogramme und activity Diagramme überarbeitet.
Ich habe den Planungs Teil weiter ergänzt.</td></tr>
<tr><td>Hilfestellungen</td><td>-</td></tr>
<tr><td>Probleme</td><td>-</td></tr>
<tr><td>Mailverkehr</td><td>
<table>
<tr><td>Zeit</td><td>Empfänger</td><td>Inhalt</td></tr>
<tr><td>08:31</td><td>Adrian Woerz</td><td>Vereinbarung des Expertenbesuches</td></tr>
</table>
</td></tr>
<tr><td>Sonstiges</td><td>Ist-Zeiten wurden eingetragen und ein Backup wurde erstellt.</td></tr>
<tr><td>Fazit</td><td>Ich bin heute gut mit der Doku vorangekommen. Ich erstellte noch ein Übersichtsdiagramm, das dem Verständnis von unbeteiligten ungemein helfen sollte.</td></tr>
</table>

Dienstag, 22.04.2014

Gesetzte Ziele	#103 Technische Dokumentation führen #701 Gespräch mit Erstexperten
Dokumentation	Ich habe heute den Implementations Teil meiner Doku weitergeführt. Ich bin gut vorangekommen.
Gespräch mit Erstexperten	Heute Morgen fand das Gespräch mit dem Erstexperten statt. Es verlief sehr gut. Wir gingen die Checkpunkte durch und stellten fest, dass soweit alles in Ordnung ist. Ich bekam nützliche Tipps zur Abgabe und zur Präsentation der IPA. Ich bekam ausserdem hilfreiche Antworten zu meinen Fragen bezüglich von Struktogrammen.
Probleme	Die Beschreibung von Diagrammen hat mehr Zeit in Anspruch genommen als erwartet.
Mailverkehr	-
Sonstiges	Ist-Zeiten wurden eingetragen und ein Backup wurde erstellt.
Fazit	Da ich die Dokumentation Zeit von Diagrammen unterschätzt hatte, kam ich heute nicht so weit wie erwartet. Dank der Einplanung der Pufferzeit ist das jedoch kein Problem. Ausserdem habe ich noch einmal den Analyse Teil überarbeitet und mit einigen nützlichen Informationen ergänzt.

Mittwoch, 23.04.2014

Gesetzte Ziele	#103 Technische Dokumentation führen
Dokumentation	Ich habe heute ein letztes Mal den Planung, Analyse und Implementation Teil überarbeitet. Ich habe Kleinigkeiten verbessert und diverse Kapitel mit weiteren nützlichen Informationen ergänzt.
Probleme	Diverse Formatvorlagen für Tabellen machten mir kleine Probleme, was jedoch sehr schnell behoben war und überhaupt nicht Zeit-kritisch war.
Mailverkehr	-
Sonstiges	Ist-Zeiten wurden eingetragen und ein Backup wurde erstellt.
Fazit	Ich konnte heute noch einmal Vollgas geben und die Zeit für die Ausbesserung der Dokumentation brauchen. Somit kann ich mich morgen vollkommen der Testdokumentation widmen.

Donnerstag, 24.04.2014

Gesetzte Ziele	#501 Testfälle ermitteln #502 Code Testen #503 Testdokumentation führen
Testfälle ermitteln	Heute habe ich alle Testfälle ermittelt. Ich war sogar schneller fertig als geplant und konnte mir somit ein wenig Reservezeit für den nächsten Task schaffen
Code Testen / Testdokumentation	Das Testen des Codes verlief ebenfalls gut. Er hat jeden Testfall erfolgreich bestanden.
Mailverkehr	-
Sonstiges	Ist-Zeiten wurden eingetragen und ein Backup wurde erstellt.
Fazit	Dank der schnellen Fertigstellung der Testfälle habe ich mir einen Vorsprung schaffen können, und hatte somit genug Zeit, der Code gründlich zu testen und zu dokumentieren.

Freitag, 25.04.2014

Gesetzte Ziele	#601 Arbeitsjournal abschliessen #602 Zeitplan abschliessen #603 Technische Dokumentation abschliessen #704 Projektabgabe
Arbeitsjournal abschliessen	Ich habe die Arbeitsjournale überprüft und abgeschlossen. Dabei hatte es noch einige Fehler in der Nummerierung der gesetzten Ziele.
Zeitplan abschliessen	Ich habe die letzten Zeiten im Zeitplan eingetragen und diesen anschliessend noch einmal kontrolliert.
Technische Dokumentation abschliessen	Ich bin noch einmal die Dokumentation durchgegangen und habe diverse Rechtschreib-, Grammatik- und Formatierungsfehler korrigiert..
Projektabgabe	Geschafft! Alle Tätigkeiten sind abgeschlossen und die IPA ist ausgedruckt und in einem Ordner zusammengefasst.
Probleme	-
Mailverkehr	-
Sonstiges	Ist-Zeiten wurden eingetragen und ein Backup wurde erstellt.
Fazit	Der letzte Tag meiner IPA ist gut verlaufen. Ich konnte alles mit einem guten Gefühl abschliessen und hatte sogar noch restliche Zeit übrig.

6. Management Summary

6.1 Ausgangslage

Zum Simulieren und Testen von grafischen Automationsaufgaben aus dem Engineering Tool CFC können die Automationspläne des Tools exportiert, und anschliessend mithilfe des Importtools IMSES in MATLAB/Simulink importiert werden. Bei gewissen Fällen können jedoch Inkonsistenten im Export auftreten.

6.2 Umsetzung

Um diese Inkonsistenzen zu beheben, wurden zuerst alle Fehlerfälle ermittelt. Danach wurde der bestehende Code und die Importdateien von IMSES analysiert. Mit diesem Wissen konnten Lösungswege geplant werden. In der Implementation wurden die Lösungen umgesetzt.

6.3 Erwartetes Ergebnis

Die Importerweiterung von IMSES ist fertig und bereit für den praktischen Einsatz. Alle Fälle von Inkonsistenzen werden behoben. Wenn Inkonsistenzen erkannt werden, erscheinen Fehlermeldungen im GUI. Neu erstellte Objekte werden, falls nötig, in einer ebenfalls neu erstellten Area abgelegt, und mit den bestehenden Objekten verlinkt, so dass die Gesamtlösung simulations- und testfähig ist.

7. Analyse

In diesem Kapitel wird die bestehende Umgebung und die vorhandene Software analysiert, um in einem nachfolgenden Kapitel die Problemlösungen effizient planen zu können.

7.1 Ziel und Zweck der Erweiterung

Der Import von ABT-Export Daten durch IMSES funktioniert grundsätzlich, ist jedoch noch fehlerbehaftet. Es kann vorkommen, dass Inkonsistenzen in den Export Daten auftreten. Diese Inkonsistenzen sollen im Rahmen der IPA durch die Programmerweiterung aufgelöst werden.

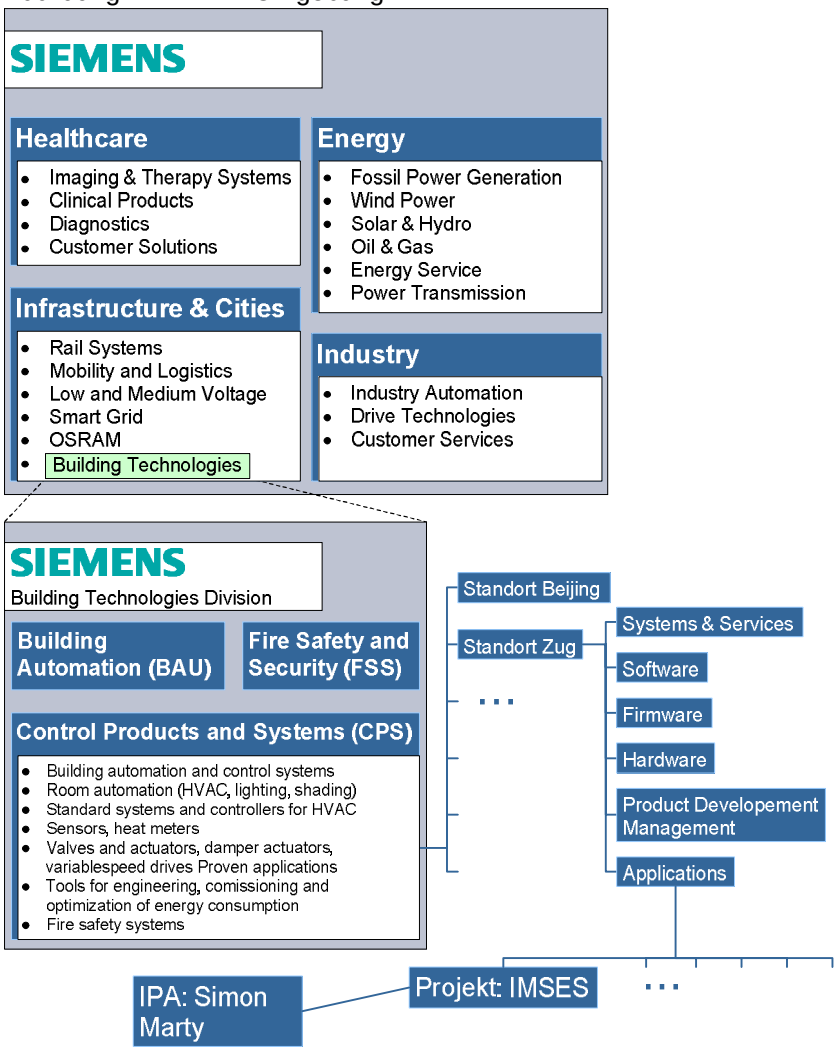
7.2 Ist Zustand

7.2.1 Bestehende Umgebung

Die Problemstellungen sollen im Rahmen eines bestehenden Programmes gelöst werden. Dieses Programm muss analysiert werden, um die möglichen und effizienten Code-Schlüsselstellen zu finden.
Bei der Siemens Building Technologies werden Applikationsfunktionen für Heizungs-, Lüftungs- und Klimakontroller in einem Engineering Tool entwickelt. Das Testen der Applikationsfunktionen mit echten Controllern und echter Hardware benötigt viel Zeit und kann kostenintensiv sein. Daher ist es sinnvoll, dass man in einer simulationsbasierten Umgebung testet. Bei der Siemens Building Technologies ist man deswegen am Aufbau einer Simulations- und Testumgebung, die sich IMSES nennt. IMSES steht dabei für ”Interface MATLAB/Simulink and Engineering System“.

Die folgende Grafik zeigt das Umfeld, in das IMSES integriert ist.

Abbildung 7–1: IPA - Umgebung



Die nachfolgende Abbildung zeigt die Schnittstelle zwischen den Export Daten des Engineering Tools und des Simulationstools Matlab/Simulink.

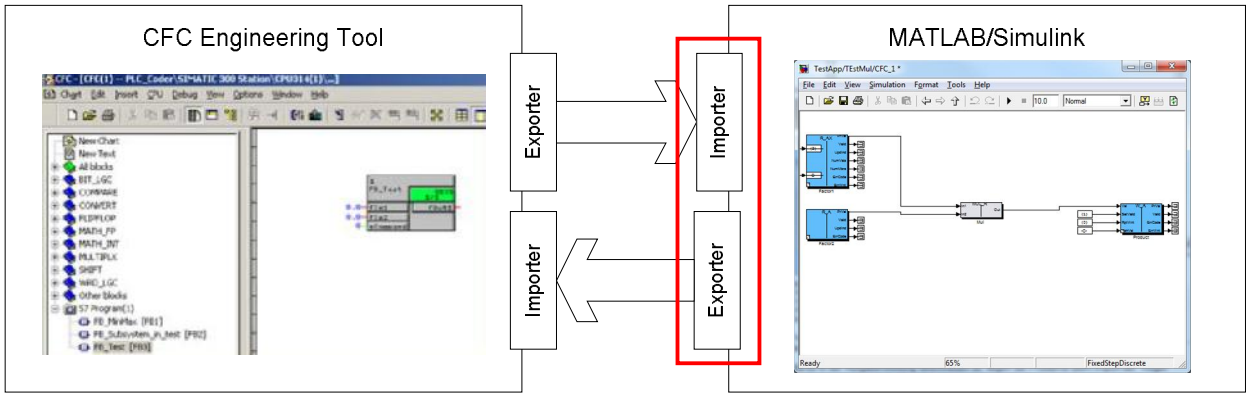


Abbildung 7–2: Einsatzbereich IMSES

Der rot markierte Bereich ist der Einsatzbereich von IMSES. Der Bereich ist im Rahmen der IPA am relevantesten, da die Erweiterungen an diesem Ort erarbeitet werden müssen.

Wie es in der Aufgabenstellung beschrieben ist, liegen die Probleme beim Import der Regel- und Steuer-Applikationen.
Wie oben beschrieben erledigt diese Aufgabe das Import Tool IMSES.

7.3 Export

Damit der Ablauf und die Funktionsweise von IMSES verständlich werden, muss klar sein, wie die Import Dateien aufgebaut und strukturiert sind.
Die Export Daten bestehen aus einer XML Datei und einer .ba Datei, wessen Aufbau identisch mit einer XML Datei ist.
Die .ba Datei enthält alle Objekte des CFC Planes.
Die XML Datei enthält die Pläne an sich, also wie die Objekte verschaltet sind, was sie für Ein-/Ausgänge haben, und inwiefern sie voneinander abhängig sind.

7.3.1.1 Aufbau Chart Datei

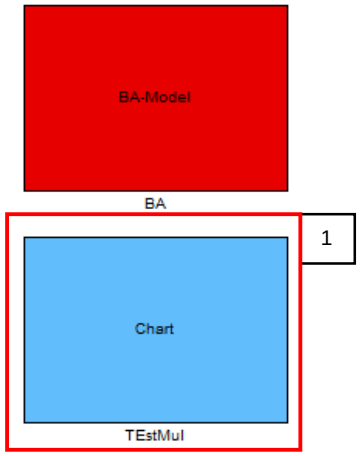
Die CFC Charts werden durch das Export Tool in eine XML Datei geschrieben.
Die XML Datei beinhaltet alle Informationen über Anordnung, Verlinkung und Existenz der verwendeten Objekte im CFC Chart.

Abbildung 7–3: Aufbau Chart: XML Ansicht

```
<AutomatedBuild xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/X
  <Chart ObjectId="121634866" Task="OB1" Name="TestApp'TEstMul@CFC_1" Comment="" Author="speckiem" Vertic
    <FB PointX="54" PointY="16" Typ="MUL_R" Name="Mul" Comment="Multiply" ItemID="0" BlockName="Mul" Bloc
      <Param Name="EN" Value="1" Invisible="true" ForTest="false" />
      <Param Name="In1" Unit="" Invisible="false" ForTest="false" Comment="" />
      <Param Name="In2" Unit="" Invisible="false" ForTest="false" Comment="" />
      <Param Name="ENO" Value="1" Invisible="true" ForTest="false" Comment="" />
      <Param Name="Out" Value="0.0" Unit="" Invisible="false" ForTest="false" Comment="" />
      <Tasks MainTaskMode="Default" />
    </FB>
    <FB PointX="9" PointY="4" Typ="R_AX" Name="Factor1" Comment="" ItemID="1">
      <Param Name="EN" Value="1" Invisible="true" ForTest="false" Comment="" />
      <Param S7_Enum="TxG_BA_2105" Name="EvlFnct" Value="2" Unit="" Invisible="false" ForTest="false" />
      <Param Name="TshVal" Value="0.1" Unit="" Invisible="false" ForTest="false" />
      <Param Name="BaObjRef" Invisible="true" ForTest="false" Comment="" />
      <Param Name="ENO" Value="1" Invisible="true" ForTest="false" Comment="" />
      <Param Name="PrVal" Value="0.0" Unit="" Invisible="false" ForTest="false" Comment="" />
      <Param Name="Valid" Value="0" Invisible="false" ForTest="false" />
      <Param Name="UpdInd" Value="0" Invisible="false" ForTest="false" />
      <Param Name="NumVals" Value="0" Unit="" Invisible="false" ForTest="false" />
      <Param Name="NumMbrs" Value="0" Unit="" Invisible="false" ForTest="false" />
      <Param S7_Enum="TxG_BA_2104" Name="ErrCode" Value="20" Unit="" Invisible="false" ForTest="false" />
      <Param S7_Enum="TxG_BA_2104" Name="ErrWrit" Value="20" Unit="" Invisible="false" ForTest="false" />
      <Param Name="RErrNr" Value="16383" Unit="" Invisible="true" ForTest="false" />
      <Param Name="WErrNr" Value="16383" Unit="" Invisible="true" ForTest="false" />
      <Tasks MainTaskMode="Default" />
    </FB>
    <FB PointX="9" PointY="17" Typ="R_A" Name="Factor2" Comment="" ItemID="2" BlockName="Factor2" BlockNa
    <FB PointX="93" PointY="16" Typ="W_A" Name="Product" Comment="" ItemID="3" BlockName="Product" BlockN
    <Link FromFBName="Mul" FromPin="Out" ToFBName="Product" ToPin="Val" Negate="false" />
    <Link FromFBName="Factor1" FromPin="PrVal" ToFBName="Mul" ToPin="In1" Negate="false" />
    <Link FromFBName="Factor2" FromPin="PrVal" ToFBName="Mul" ToPin="In2" Negate="false" />
    <Interface DataType="Bool" Name="EN" Comment="" SectionType="VarInput" Value="1" Unit="" Invisible="f
  </Chart>
</AutomatedBuild>
```

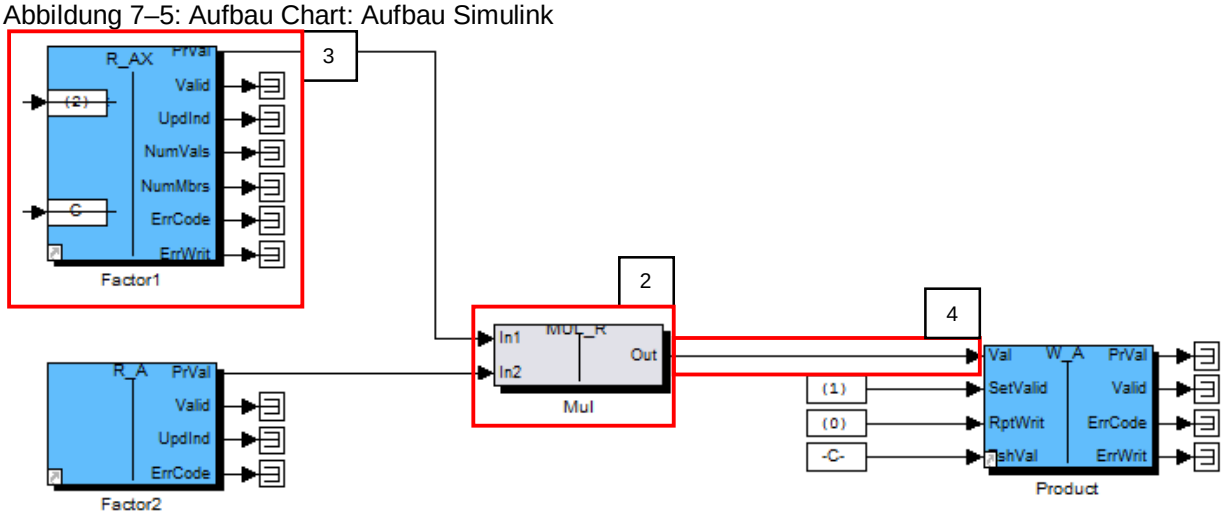
Der ganze Chart enthält alle Pläne, Objekte und Funktionen. Die Abbildung zeigt die Grundansicht von den BA Objekten und den Charts.

Abbildung 7–4: Aufbau Chart: Simulink Ansicht



Wenn man in den Chart rein springt, erscheint die Ansicht folgender Abbildung.

Abbildung 7–5: Aufbau Chart: Aufbau Simulink



Die Bereiche in der XML Ansicht entsprechen den Elementen in der Simulink Ansicht. Folgende Tabelle beschreibt die nummerierten Bereiche.

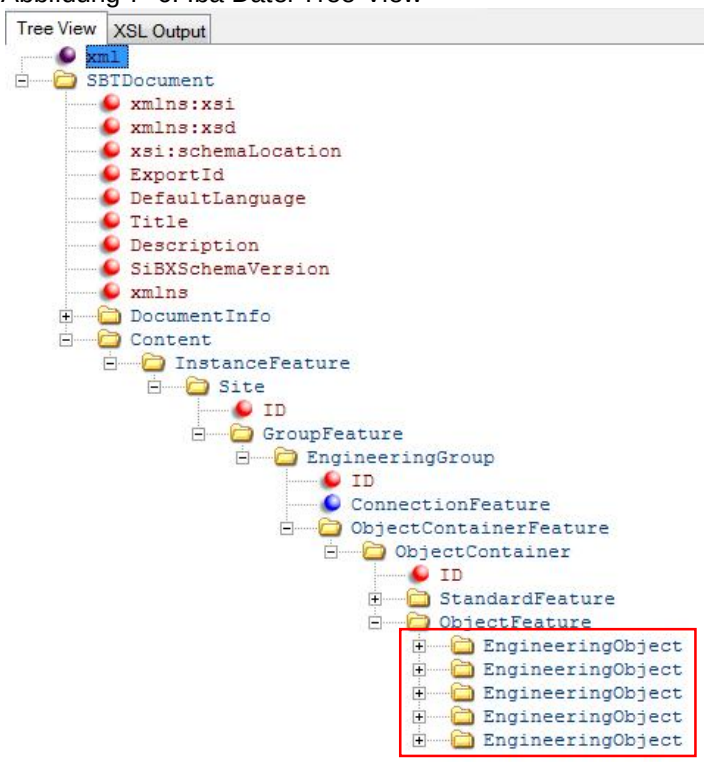
Tabelle 7–1: XML Aufbau Beschreibung

Nr.	Beschreibung
1	Ganzer Chart mit allen Blöcken und Verbindungen
2	Block namens “Mul”. Hat 2 Eingänge und ein Ausgang. Multipliziert beide Zahlen an den Eingängen und gibt sie als Resultat aus
3	Block namens “Factor1”. Wie der Name es andeutet, repräsentiert der Block der erste Faktor für die Multiplikation
4	Verbindung von “Mul” nach “Product”. Über diese Verbindung wird das Resultat von “Mul” nach “Product” geschrieben.

7.3.1.2 Aufbau BA Datei

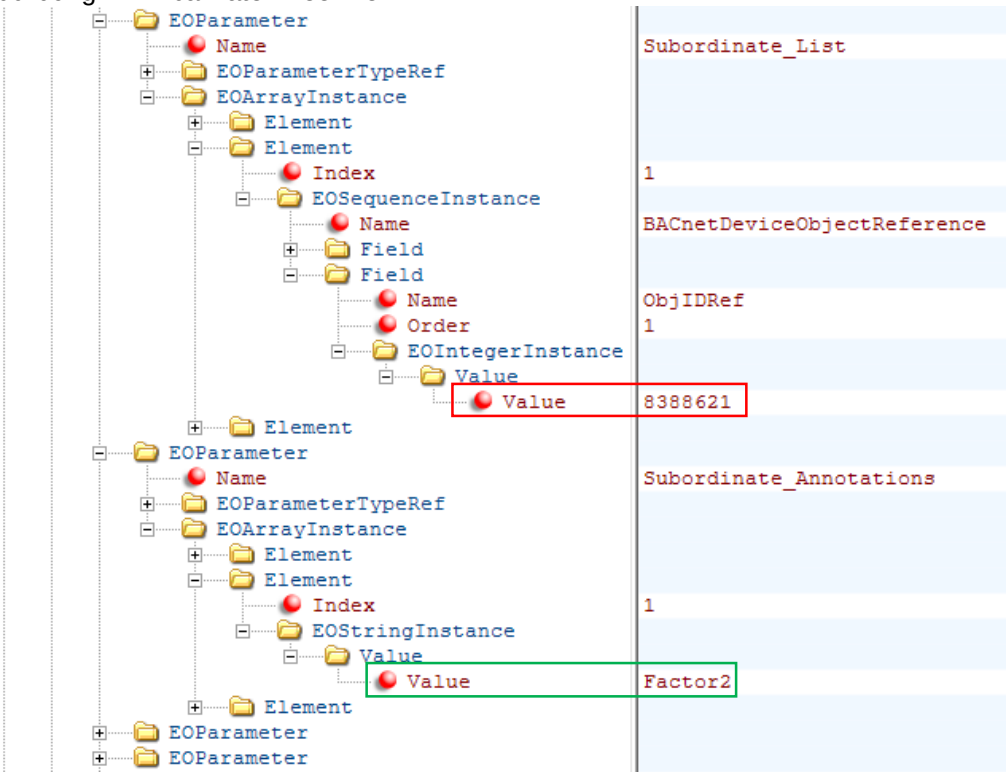
In der nachfolgenden Abbildung sieht man die Struktur einer .ba Datei. Wichtig sind vor allem die rot markierten Verzweigungen namens „EngineeringObject“.
Diese Objekte werden durch das Import Tool IMSES rekursiv ausgelesen, und in ein Simulink Plan importiert. Warum die Objekte rekursiv ausgelesen werden, wird weiter unten beschrieben.

Abbildung 7–6: .ba Datei Tree-View



In einem Objekt findet man viele Parameterwerte. Im Rahmen der IPA sind vor allem die Parameter „Subordinate_List“ und „Subordinate_Annotations“ von Bedeutung.
Beide Parameter sind Arrays.
Die Subordinate_List enthält ID’s von Child Objekten, welche als Verweise zu den Objekten fungieren (in der Abbildung unten rot markiert).
Die Subordinate_Annotations enthält Platzhalter für die Elemente in der Subordinate_List, welche die Namen der Child Objekte repräsentieren (in der Abbildung unten grün markiert).
Die Elemente in der Subordinate_List und in der Subordinate_Annotations sind äquivalent angeordnet.
Somit gehört die ID mit Index 1 von der Subordinate_List zu dem Platzhalter mit Index 1 von Subordinate_Annotations.

Abbildung 7–7: .ba Datei Tree-View



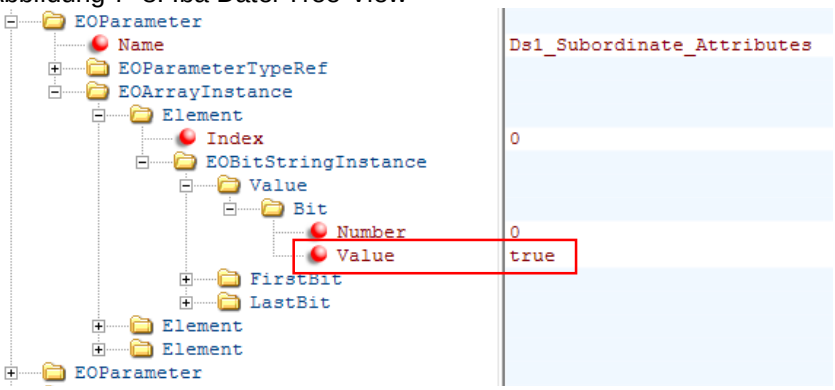
Jedes Objekt in der Subordinate_List besitzt den Status „Connected“ oder „Owned“.

- Wenn es „Owned“ ist, muss das Objekt im .ba Datei vorhanden sein
- Wenn es “Connected” ist, ist es nicht im aktuellen Modul enthalten, sondern in einem anderen

Der Parameter „Ds1_Subordinate_Attributes“ enthält diese Information. Auch dieser Parameter ist ein Array, welches für jedes Objekt in Subordinate_List die Statusinformation enthält. In der Abbildung sieht man der relevante Wert.

- “True” steht für “Connected”
- “False” steht für “Owned”

Abbildung 7–8: .ba Datei Tree-View

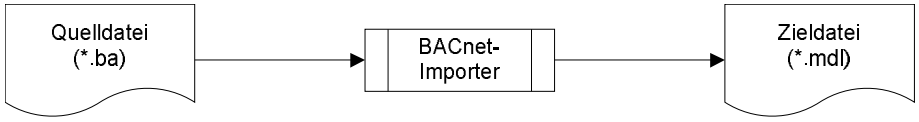


7.3.2 Tool Analyse

Das Import Tool IMSES muss analysiert werden, damit klar ist, wo Änderungen vorzunehmen sind und welche Funktionalitäten bereits vorhanden sind.

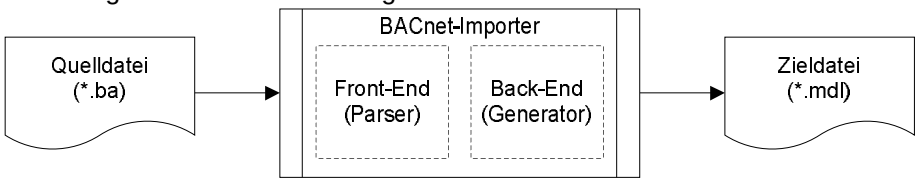
Als Import Input dient die Quelldatei des Typs ba.
Eine solche ba Datei ist identisch aufgebaut wie eine XML Datei.
Die Quelldatei wird durch den BACnet-Importer (IMSES) importiert. Aus dem Import wird schlussendlich eine .mdl Datei erzeugt, welche vom Simulationstool Simulink gelesen und geöffnet werden kann.

Abbildung 7–9: Import Ablauf - IMSES



Der BACnet-Importer besteht aus zwei Bereichen. Zum Einen das Front-End, und zum Anderen das Back-End.
Das Front-End ist für das Parsing, also für das Auslesen der .ba Datei zuständig.
Das Back-End generiert Simulink-Objekte der zuvor ausgelesenen .ba Daten.
In dieser Grafik wird der ganze Ablauf verdeutlicht.

Abbildung 7–10: Funktionsteilung - IMSES



Das Front-End besteht aus einer Hauptfunktion „parse“, welche die Sub-Funktionen „getSubList“ und „generateObjectList“ enthalten.
Die Funktion getSubList liest alle Daten der untergeordneten Child-Objekten eines EngineeringObjects (siehe Abbildung 7–6: .ba Datei Tree-View, rote Markierung) der .ba Datei aus.

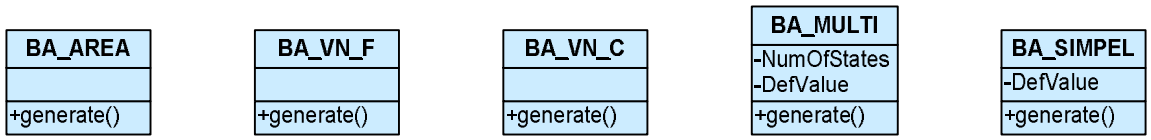
Zur Veranschaulichung ein Element in der SubList:

Abbildung 7–11: SubList Beispiel

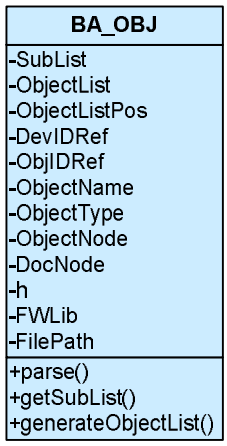
SubList{1,1}		
1	4194303	Device ID
2	20971523	Object ID (Verweis)
3	PscDetRs	Object Name (Platzhalter)
4	true	True = Owned False = Connected

Diese Objekte werden an die Funktion generateObjectList übergeben. Jedes Objekt wird aufgrund des Objekt Typs einer Matlab-Klasse zugewiesen.
Im Folgenden ein Überblick der Objekt Klassen:

Abbildung 7–12: Objekt Klassen



Alle Klassen sind von der Klasse BA_OBJ abgeleitet.



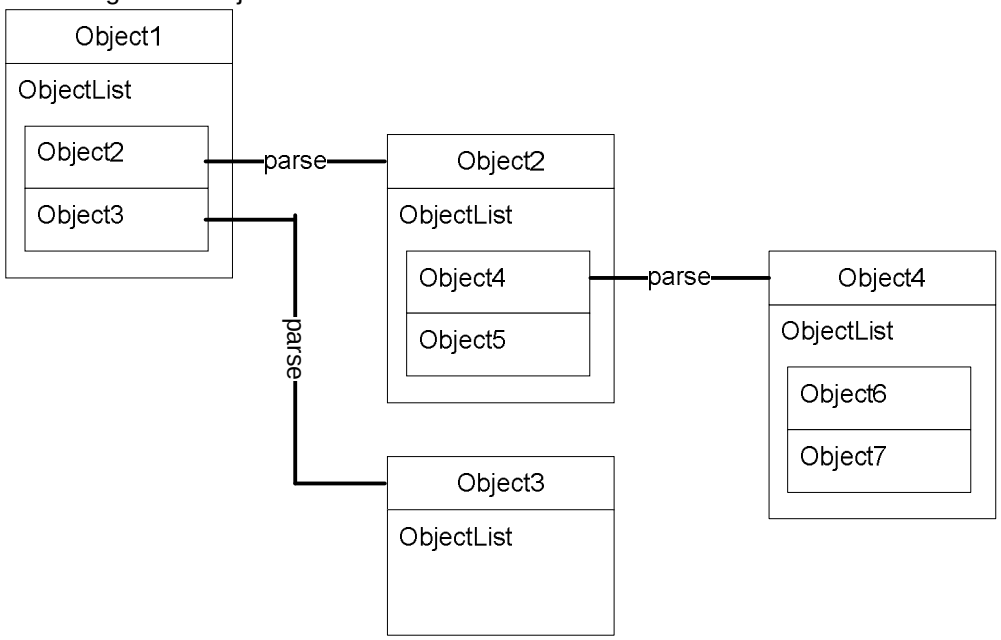
Das Objekt wird erzeugt und in ein Array ObjectList eingefügt.
Ein Beispiel der ObjectList:

Abbildung 7–13: ObjectList - Beispiel

ObjectList{}	
0	BA_AREA
1	BA_VN_F
2	BA_SIMPEL
3	BA_SIMPEL

Jedes dieser Objekte ist ein Child Objekt. Diese Child Objekte können wieder Child Objekte enthalten, und jene Child-Child Objekte wieder.
Da die Child-Objekte in der ObjectList alle Funktionen von der Hauptklasse BA_OBJ erben, besitzt jedes von ihnen die Funktion parse().

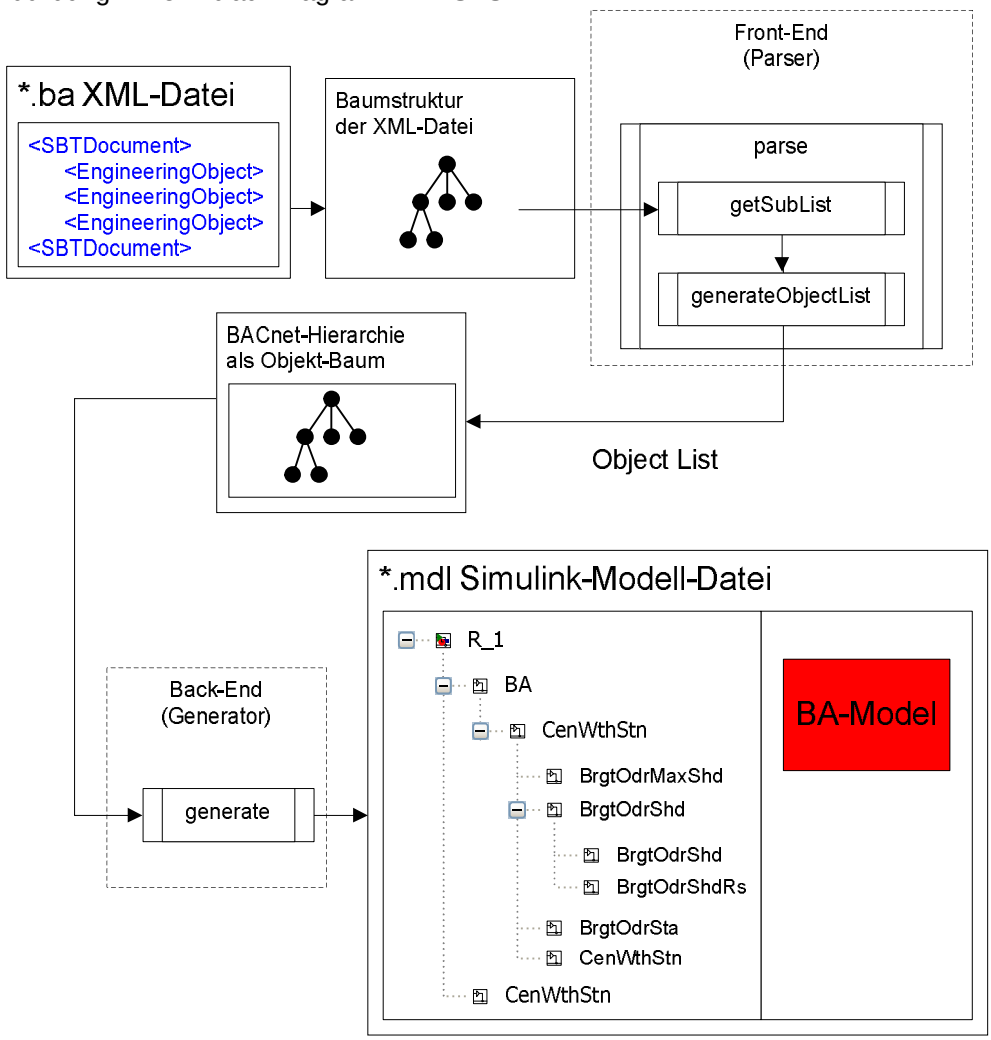
Abbildung 7–14: Objekt Hierarchie



Nachdem die ObjectList erzeugt ist, wird von jedem Objekt in ihr die parse Funktion ausgeführt. Mit der Ausführung der parse Funktion beginnt das Ganze wieder von vorne.
Die SubList und ObjectList wird vom Child-Objekt ausgelesen, die parse Funktion von den Objekten in der objectList des Child-Objekts wird ausgeführt usw.
Solange, bis das Ende des Hierarchie Baumes erreicht wurde. Schlussendlich wird dann von jedem Objekt die generate (siehe Abbildung unten) Funktion ausgeführt, um das Objekt in Simulink zu erzeugen.
So bildet sich eine Hierarchie aus Objekten.

Um der Ganze Ablauf übersichtlich zu gestalten, wurde ein Diagramm erstellt

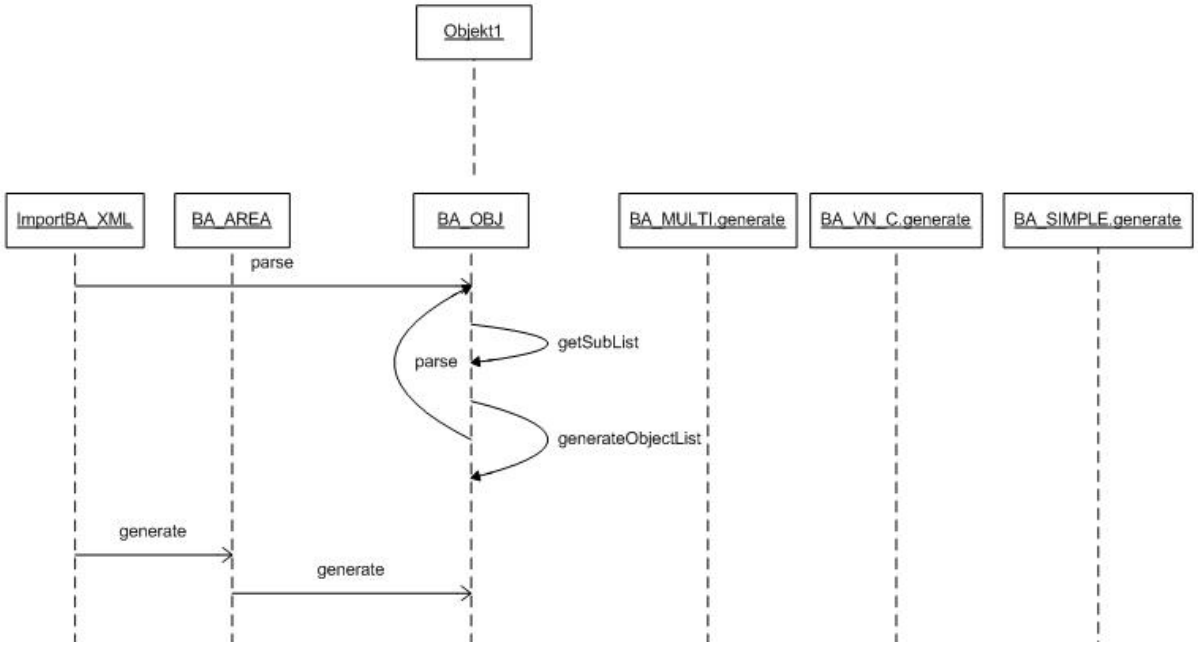
Abbildung 7–15: Ablauf Diagramm - IMSES



7.3.3 Sequenzdiagramm

Um den zeitlichen Ablauf des ganzen Vorgangs darzustellen, wurde ein Sequenzdiagramm erstellt.

Abbildung 7–16: Sequenzdiagramm



7.4 Soll Zustand

Die Importfunktion soll so angepasst werden, dass alle Fehlerfälle abgedeckt werden und entsprechende Massnahmen durchgeführt werden. Fehlende Objekte sollen in SIMULINK erstellt und korrekt verlinkt werden. Vor der Implementierung muss klar sein, wo der neue Code platziert wird, ob neue Klassen entstehen oder neue Funktionen erstellt werden müssen.
Um die Fälle 1-3 abzudecken, müssen folgende Funktionalitäten vorhanden sein:

- Auslesen einer .ba Datei
- Auslesen einer Chart Datei
- Erzeugung von neuen Objekten
- Bearbeitung von neuen Objekten

8. Planung

8.1 Fehlersituationen

Die Importdaten können unvollständig oder Inkonsistenz sein. In welchen Fällen solche Inkonsistenzen vorkommen können, wird in diesem Kapitel beschrieben. Ausserdem werden mögliche Lösungen dargestellt, welche mit Activity Diagrammen veranschaulicht werden.

8.1.1 Fall 1

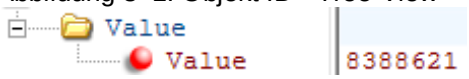
In der .ba Datei besitzt ein Objekt ein Verweis auf ein Child-Objekt. Das Child Objekt ist jedoch nicht in der .ba Datei und nicht im CFC Plan(Chart XML Datei) vorhanden.
Dieses Problem führt zu einer Inkonsistenz in dem Modul.
Damit diese Inkonsistenz behoben werden kann, muss das fehlende Objekt erstellt werden.
Um das Objekt zu erstellen, ist es notwendig die Objekt Typ ID zu kennen.
Da jedoch kein Objekt vorhanden ist, welches die benötigte Information liefern würde, muss die Objekt Typ ID anhand des Verweises berechnet werden.

Die Regel für die Zusammensetzung einer Objekt Typ ID lautet wie folgt:

Verweis = Objekt Typ ID * 2 ^ 22

Beispiel für die Umrechnung:

Abbildung 8–1: Objekt ID - Tree-View

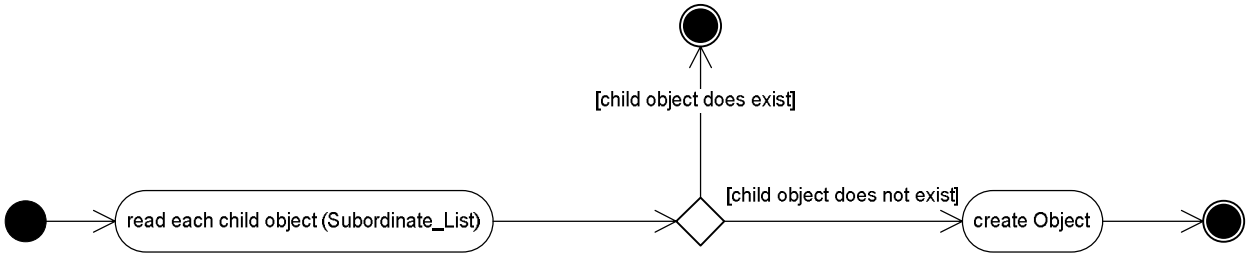


Objekt Typ ID = Verweis / 2 ^ 22
→ Objekt Typ ID = 8388621 / 2 ^ 22 = 2.000003099

Da das Ergebnis nur als Ganzzahl gebraucht werden kann, wird das Ergebnis auf die nächste niedrigere Ganzzahl gerundet.

Anhand der Objekt Typ ID kann nun die generate Funktion der zugehörigen Klasse ausgeführt werden.

Abbildung 8–2: Activity Diagramm - Fall 1

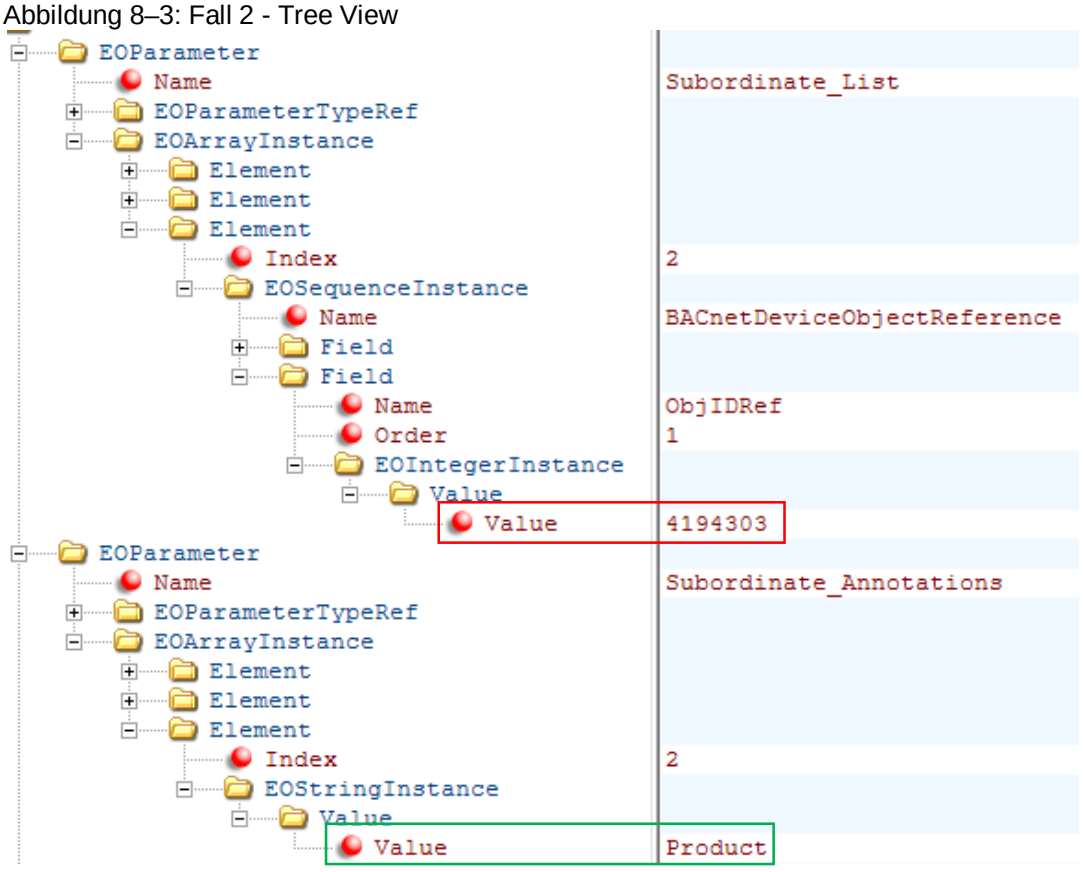


8.1.2 Fall 2

In der .ba Datei besitzt ein Objekt ein Platzhalter für einen Verweis auf ein Child-Objekt, jedoch existiert dieser Verweis auf ein Child-Objekt nicht. Das Objekt wird jedoch im CFC Plan (Chart XML Datei) verwendet.
Wenn das Objekt „Connected“ ist, ist es nicht im zu testenden Modul enthalten sondern in einem anderen.
Wenn es „Owned“ ist, besteht eine Inkonsistenz im zu testenden Modul, eine Fehlermeldung an den Anwender erfolgt.
Das Objekt muss in beiden Fällen erstellt werden.

8.1.2.1 Beispiel

Abbildung 8–3: Fall 2 - Tree View



Der Verweis auf das Child-Objekt ist eine ID bestehend aus einer Ganzzahl mit dem Wert 4194303 (in der Abbildung rot markiert). Der ID Wert 4194303 ist ein default Wert und bedeutet, dass die ID kein valider Verweis ist.
Es ist jedoch ein Platzhalter für diesen Verweis vorhanden (in der Abbildung grün markiert).

Im XML Chart File ist das Child-Objekt vorhanden, das erkennt man an der Namensbezeichnung (in der Abbildung rot markiert). Deshalb ist es notwendig, das Objekt zu erstellen.
Wie im Analyse Kapitel beschrieben, besitzt jede Objekt-Klasse eine generate Funktion. Das Erzeugen eines Objektes in Simulink erfordert einen Objekt-Typ, damit die generate Funktion der richtigen Klasse ausgeführt wird. Der Objekttyp wird anhand der Objektbezeichnung durch ein Regelwerk bestimmt.

Anhand der Objekt Bezeichnung in der Spalte „Name“ (in der Grafik rot markiert) wird entschieden, welche Objekt Typ ID verwendet werden muss. Diese steht in der dritten Zeile in der Grafik(grün markiert).

Abbildung 8–4: Ausschnitt XML Chart File

```
<FB PointX="93" PointY="16" Typ="W_A" Name="Product"
  <Param Name="EN" Value="1" Invisible="true" ForTest
  <Param Name="Val" Unit="" Invisible="false" ForTest
  <Param Name="SetValid" Value="1" Invisible="false"
  <Param Name="RptWrit" Value="0" Invisible="false" F
  <Param Name="TshVal" Value="0.1" Unit="" Invisible=
  <Param Name="BaObjRef" Invisible="true" ForTest="fa
  <Param Name="ENO" Value="1" Invisible="true" ForTes
  <Param Name="PrVal" Value="0.0" Unit="" Invisible="
  <Param Name="Valid" Value="0" Invisible="false" For
  <Param S7_Enum="TxG_BA_2104" Name="ErrCode" Value="
  <Param S7_Enum="TxG_BA_2104" Name="ErrWrit" Value="
  <Param Name="RErrNr" Value="16383" Unit="" Invisibl
  <Param Name="WErrNr" Value="16383" Unit="" Invisibl
  <Tasks MainTaskMode="Default" />
</FB>
```

In der Beispiel-Abbildung oben lautet der Objekt Name „W_A“.
In diesem Fall ist als Objekt Typ ID 2 zu verwenden (in der nachfolgenden Abbildung blau markiert).

Abbildung 8–5: Ausschnitt des Regelwerks

Category		Datatype	Name	Compatible BACnet Objects						
				ACalcVal	ACnfVal	AI	AO	APrcVal	BCalcVal	BcnfVal
				ObjType = 2	2	0	1	2	5	5
Read	Single	Real	R_A	x	x	x	x	x		
		Boolean	R_B						x	x
		Unsigned	R_M							
		Unsigned	R_UNSG							
		Structure	R_LGTCMD							
		Structure	R_BLSCMD							
	Multiple	Real	R_AX	x	x	x	x	x		
		Boolean	R_BX						x	x
		Unsigned	R_MX							
		Structure	R_LGTX							
		Structure	R_BLSX							
Write	Single	Real	W_A	x						
		Boolean	W_B						x	
		Unsigned	W_M							
Command	Single	Real	CMD_A				x	x		
		Boolean	CMD_B							
		Unsigned	CMD_M							
		Structure	CMD_LGT							
		Structure	CMD_BLS							
	Multiple	Real	CMD_AX				x	x		
		Boolean	CMD_BX							
		Unsigned	CMD_MX							
Read Property	Single	Real	RP_REAL	Allows access to any BACnet object property of datatype Real.						
		Boolean	RP_BOOL	Allows access to any BACnet object property of datatype Boolean or BACnetBinary						
		Unsigned	RP_UNSG	Allows access to any BACnet object property of datatype Unsigned (includes Multi						
Write Property	Single	Real	WP_REAL	Allows access to any writable BACnet object property of datatype Real.						
		Boolean	WP_BOOL	Allows access to any writable BACnet object property of datatype of Boolean or BA						
		Unsigned	WP_UNSG	Allows access to any writable BACnet object property of datatype Unsigned (includ						
Collecting	Group	Structure	R_AG	Allows access to any BACnet object property of datatype Real.						
		Structure	R_BG	Allows access to any writable BACnet object property of datatype of Boolean or BA						
		Structure	R_MG	Allows access to any BACnet object property of datatype Unsigned (includes Multi						
Distributing	Group	Structure	CMD_AG	Allows access to any BACnet object property of datatype Real.						
		Structure	CMD_BG	Allows access to any writable BACnet object property of datatype of Boolean or BA						
		Structure	CMD_MG	Allows access to any BACnet object property of datatype Unsigned (includes Multi						
		Structure	CMD_LGTG	Allows access to analog or binary BACnet lighting output objects.						
		Structure	CMD_BLSG	Allows access to BACnet blind output objects.						

Für das Herausfinden der Objekt Typ ID werden zwei Ansätze weiterverfolgt.
Dafür werden beide Varianten analysiert und deren Vor- und Nachteile aufgeführt.
Anschliessend wird das schlussendliche Vorgehen aus den beiden Varianten evaluiert.

8.1.2.2 Variante 1

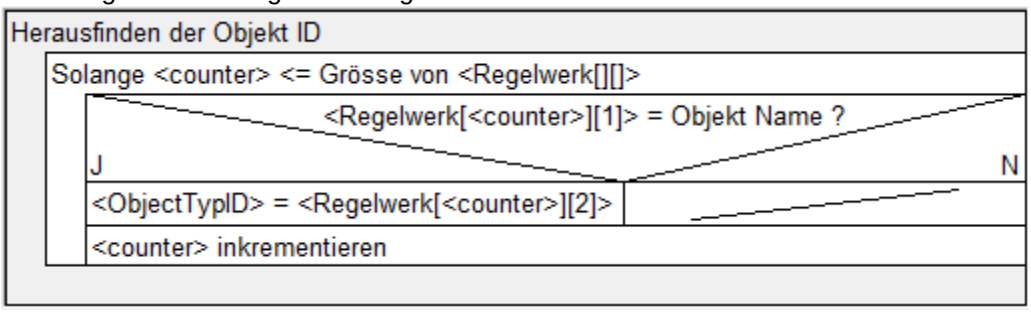
Bei dieser Variante würde das Regelwerk fest in den bestehenden Code von IMSES integriert werden.
Da nur eine Informationen zum Herausfinden der Objekt Typ ID benötigt wird, nämlich der Objekt Name, liesse sich das Regelwerk in ein zweidimensionales Array integrieren.

Tabelle 8–1: Regelwerk in Array - Beispiel

Index	Name	Objekt Typ ID
0	R_A	2
1	R_B	5
2	R_M	19

Wenn ein neues Objekt erstellt werden müsste, und die Objekt Typ ID erforderlich wäre, wäre es der einfachste Weg folgender Lösungsansatz zu verwenden:

Abbildung 8–6: Struktogramm: Regelwerk



Diese Variante wäre sehr effizient und schnell umgesetzt. Es wären wenige Code Zeilen, um die das Tool ergänzt werden müsste. Wenn jedoch das Regelwerk zu einem späteren Zeitpunkt geändert werden sollte, muss der Code angepasst werden, was nicht unbedingt benutzerfreundlich wäre.

8.1.2.3 Variante 2

Bei dieser Variante wird eine eher umständliche Lösung beschrieben, welche jedoch flexibler und anpassbarer wäre als die vorherige Variante.
Das Regelwerk wäre in einer .XML, .csv, oder Excel Datei untergebracht.
Die Datei müsste von Matlab ausgelesen werden. Dies wäre relativ zeitintensiv und der Code wäre unübersichtlich in die Gesamt-Umgebung integriert.
Es wäre jedoch wie gesagt die flexiblere Lösung als Variante 1, da bei einer Änderung des Regelwerks einfach die Datei angepasst werden könnte, und nicht das Tool an sich verändert werden müsste.

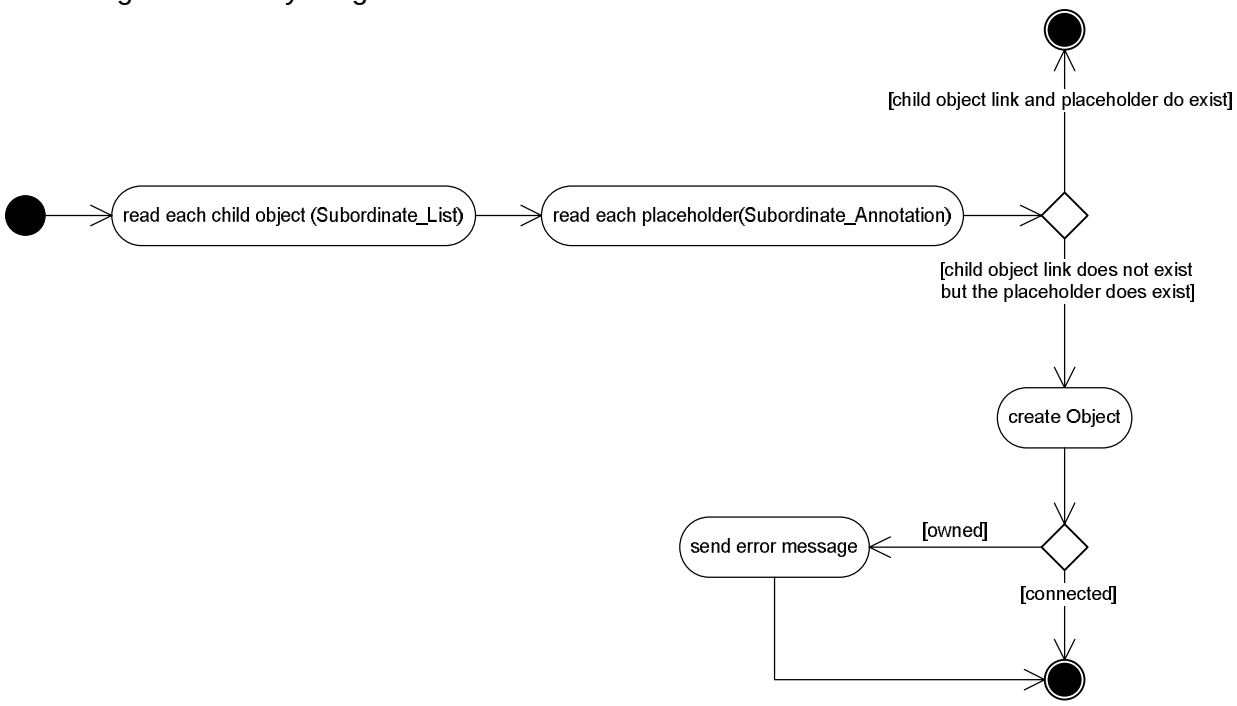
8.1.2.4 Variantenanalyse

Tabelle 8–2: Regelwerk integrieren Evaluation

Beurteilungskriterien	Produkte	Gewichtung	Ergebnisse	Punkte(max. 10)	Resultat
Übersichtlichkeit im Code	Variante 1	30	sehr gut	8	240
	Variante 2		mittel	6	180
Realisierbarkeit (im Umfang der IPA)	Variante 1	60	sehr gut	10	600
	Variante 2		mittel	4	240
Anpassbarkeit	Variante 1	10	schlecht	2	20
	Variante 2		Sehr gut	10	100
Variante 1	**840**				
Variante 2	**520**				

Die Variantenanalyse zeigt, dass die Variante 1 besser geeignet ist. Variante 2 wäre von der Anpassbarkeit deutlich besser geeignet. Die Anpassbarkeit wird jedoch weniger stark gewichtet, da das Regelwerk selten bis nie verändert wird.

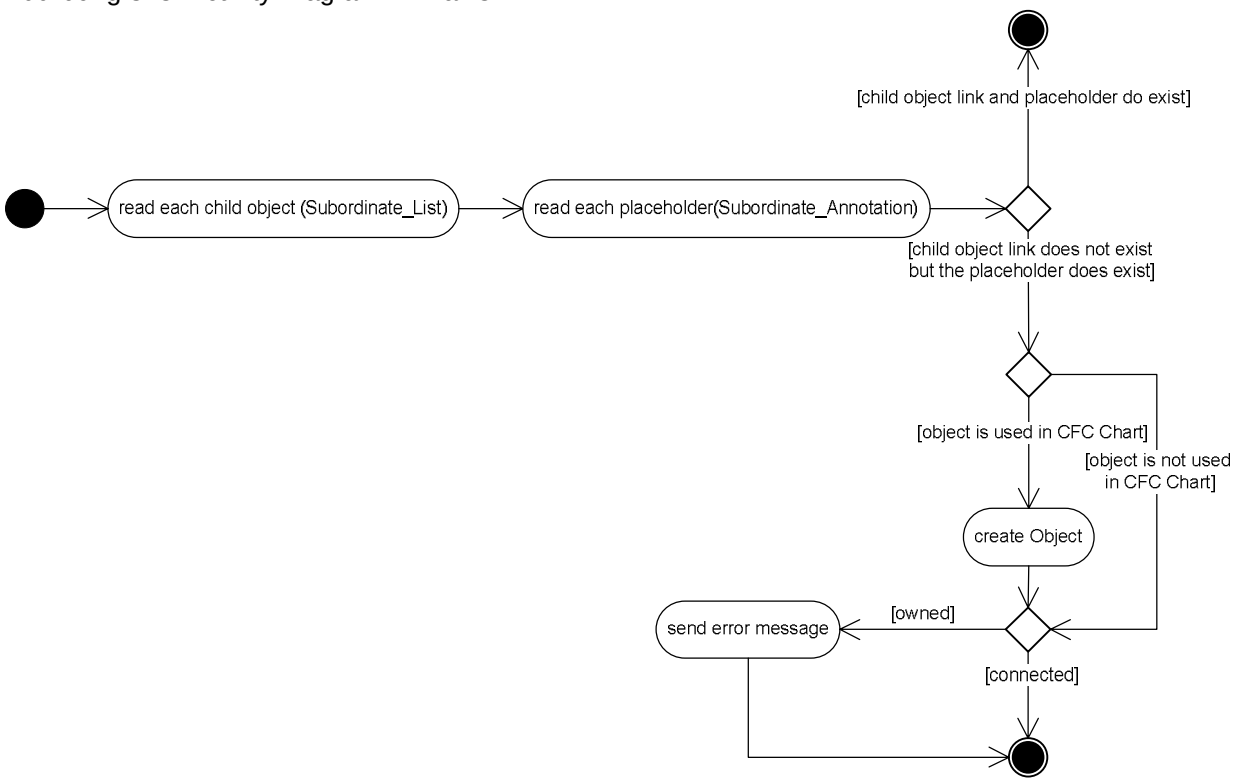
Abbildung 8–7: Activity Diagramm - Fall 2



8.1.3 Fall 3

Fall 3 ist gleich wie Fall 2, ausser dass das Objekt NICHT im CFC Plan (Chart XML Datei) verwendet wird. Somit hat das Objekt keine Funktion, sondern ist reine Information.
Es muss kein Objekt erstellt werden. Wenn das Objekt „Owned“ ist, erfolgt eine Fehlermeldung an den Anwender.

Abbildung 8–8: Activity Diagramm - Fall 3



9. Implementierung & Umsetzung

In diesem Kapitel werden alle Schlüsselstellen, Code Abschnitte und sonstige Aktivitäten aufgeführt, welche im Umfang der IPA erarbeitet und umgesetzt wurden.
Auszüge aus dem Quellcode sind im Anhang aufgeführt.
Ein grosser Teil der Projektarbeit war es, das bestehende Tool zu analysieren und die Lösung zu planen. Dies war auch der anspruchsvolle Teil des Projekts. Als es klar war, welchen bereits bestehenden Code weiterverwendbar ist, und wo die Schlüsselstellen sind, waren 90% der Implementation bereits geschafft.
Siehe dazu die Kapitel Analyse & Planung.
Der eigentliche Code der zur Problemlösung notwendig war, ist daher nicht von grossem Umfang.

9.1 Struktogramme

Struktogramme sind übersichtlicher und einfacher zu lesen als reiner Programmcode.
Darum wurden zur Darstellung des Code Aufbau & Ablauf Struktogramme zu Fall 1 und Fall 2 erstellt.

Abbildung 9–1: Struktogramm Fall 1

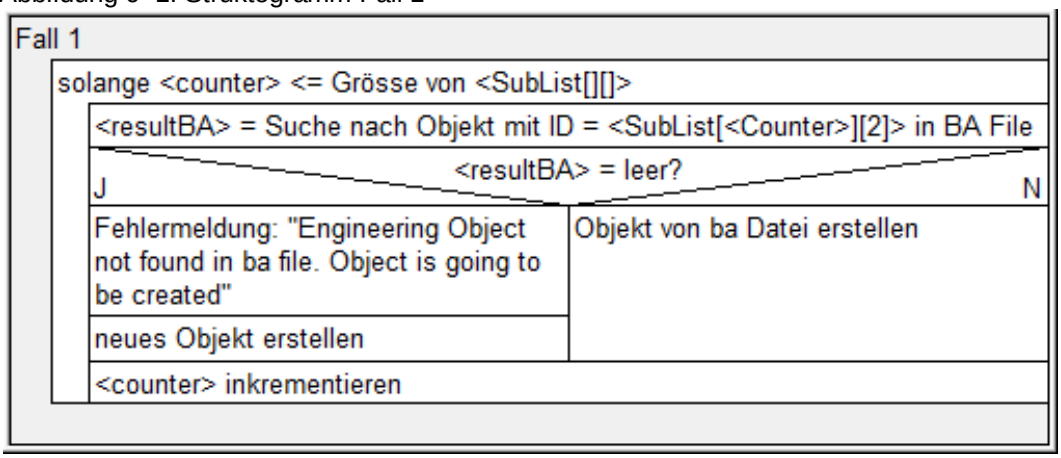
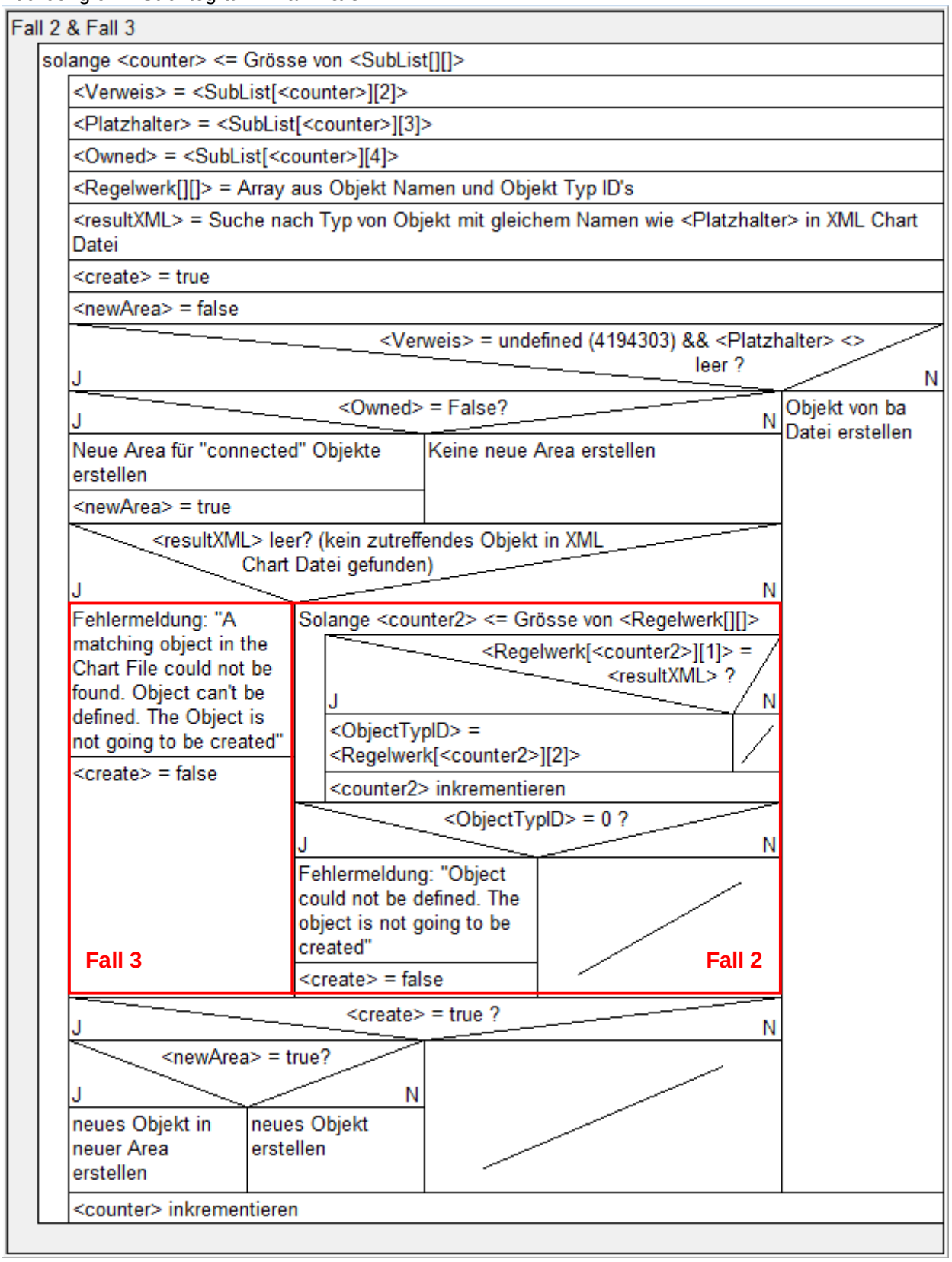


Abbildung 9–2: Struktogramm Fall 2 & 3



9.2 getXmlChartName

Tabelle 9–1: Funktionsbeschreibung

Name	getXmlChartName
Parameter	obj (aktuelles Objekt, gleich wie „this“)
Rückgabewert	xmlChartName
Beschreibung	liest den Chart File Name vom Full-Import ZIP File aus und gibt ihn als Funktionswert zurück. Wird verwendet, um später die XML Datei öffnen zu können

```
function xmlChartName = getXmlChartName(obj)
   % liest den Chart File Name aus und gibt ihn als Funktionswert zurück
   persistent ChartName;
   if isempty(ChartName)
       % XML File vom Full Import ZIP File
       xmlChart = dir([obj.FilePath '\*.XML']);
       ChartName = xmlChart.name;
   end
   xmlChartName = ChartName;
end
```

9.3 getUniqueObjID

Tabelle 9–2: Funktionsbeschreibung

Name	getUniqueObjID
Parameter	keine
Rückgabewert	ObjID
Beschreibung	Erzeugt bei jedem Aufruf eine eindeutige ID. Beginnt bei 11000. Die ID wird bei jedem Aufruf inkrementiert. Wird neu erstellten Objekten zugewiesen

```
function ObjID = getUniqueObjID
   % eindeutige ID erzeugen
   % wird als ObjID für neu erzeugte Objekte gebraucht
   persistent ID;
   if isempty(ID)
       ID=11000;
   end
   ID=ID+1;
   ObjID = int2str(ID);
End
```

Definition persistent:

„Persistent variables are local to the function in which they are defined, but they retain their values in memory between calls to the function.“

Quelle: http://www.mathworks.ch/ch/help/simulink/ug/defining-and-initializing-persistent-variables.html

9.4 XML Chart Datei

Mit der Funktion getXmlChartName (siehe weiter oben) wird der Funktion xmlread der XML Chart Dateiname übergeben, welche die XML Datei als Java xPath Objekt der Variable xDoc zuweist.

```
% XML Object vom Chart File
xDoc = xmlread(obj.getXmlChartName(obj));
…
```

Mit xPath wird nach jedem Knoten in der XML Datei gesucht, der „FB“ heisst und dessen „Name“ Attribut gleich dem Platzhalter des aktuellen Child-Objektes ist. Wenn dieser Knoten gefunden wurde, wird der Wert des Attributes „Typ“ in die Variable resultXML geschrieben.

```
% Chart File nach Sublist Objektnamen durchsuchen
exprXML=xpath.compile(['.//FB[@Name="' obj.SubList{3,k} '"]/@Typ']);
resultXML=exprXML.evaluate(xDoc, XPathConstants.STRING);
```

Wenn resultXML leer ist, bedeutet das, dass das fehlende Objekt nicht i der XML Chart Datei vorhanden ist. (siehe Struktogramm Fall 3)

9.5 Implementierung Regelwerk

Das Regelwerk wird beim Programmstart mit allen Werten initialisiert.
Wie im Kapitel Planung beschrieben, ist die Lösung statisch und nur im Code anpassbar.

```
% Initialisiere BA Obj Types
obj.BAObjTypeNames = {'R_A' 2;'R_B' 5;'R_M' 19;'R_UNSG' 48;...
                      'R_LGTCMD' 260;'R_BLSCMD' 258;'R_AX' 2;...
                      'R_BX' 5;'R_MX' 19;'R_LGTX' 262;'R_BLSX' 259;...
                      'W_A' 2;'W_B' 5;'W_M' 19;'CMD_A' 2;'CMD_B' 5;...
                      'CMD_M' 19;'CMD_LGT' 260;'CMD_BLS' 258;...
                      'CMD_AX' 2;'CMD_BX' 5};
```

Die Auswahl der Objekt Typ ID (Variablenname im Code: „ProtoType“) des Regelwerks:

```
% wenn Object Type im Chart gefunden wurde
for c=1:size(obj.BAObjTypeNames)
    if strcmpi(obj.BAObjTypeNames(c,1),resultXML)
        % Object type Nummer von Regelwerk herauslesen
        ProtoType=obj.BAObjTypeNames{c,2};
    end
end
```

9.6 Erzeugung eines Objektes

Bei den Fällen 1 & 2 muss ein neues Objekt erstellt werden. Diese Funktionalität ist bereits in IMSES vorhanden (Siehe dazu Kapitel Analyse).
Alle Elemente in der ObjectList sind Objekte welche über eine generate Funktion verfügen. Wenn also ein neues Objekt erstellt werden soll, muss es der ObjectList hinzugefügt werden, alles Weitere wird von IMSES erledigt.

Codezeile, welche der objectList des aktuellen Objektes ein neues Objekt, konstruiert von der Klasse BA_SIMPEL, hinzufügt.

```
obj.ObjectList{count}=BA_SIMPEL;
```

10. Testing

Im folgenden Kapitel werden die relevanten Testfälle für die Erweiterung des Importes von IMSES ermittelt und durchgeführt.

10.1 Testumgebung

Die Durchführung der Tests findet in der folgenden Umgebung statt:
(Ausgelesen aus dem Microsoft Hilfsprogramm „msinfo32“)

Abbildung 10–1: msinfo32 System Summary

Betriebssystemname	Microsoft Windows 7 Enterprise
Version	6.1.7601 Service Pack 1 Build 7601
Zusätzliche Betriebssystembesc...	Nicht verfügbar
Betriebssystemhersteller	Microsoft Corporation
Systemname	MD19YMYC
Systemhersteller	FUJITSU
Systemmodell	LIFEBOOK E780
Systemtyp	x64-basierter PC
Prozessor	Intel(R) Core(TM) i5 CPU M 520 @ 2.40GHz, 2400 MHz, 2 Kern(e), 4 logis...
BIOS-Version/-Datum	FUJITSU // Phoenix Technologies Ltd. Version 1.20, 24.01.2011
SMBIOS-Version	2.6
Windows-Verzeichnis	C:\WINDOWS
Systemverzeichnis	C:\WINDOWS\system32
Startgerät	\Device\HarddiskVolume1
Gebietsschema	Vereinigte Staaten von Amerika
Hardwareabstraktionsebene	Version = "6.1.7601.17514"
Benutzername	WW002\z002t6fy
Zeitzone	Mitteleuropäische Sommerzeit
Installierter physikalischer Speic...	6.00 GB
Gesamter realer Speicher	5.80 GB
Verfügbarer realer Speicher	3.52 GB
Gesamter virtueller Speicher	11.6 GB
Verfügbarer virtueller Speicher	9.07 GB
Größe der Auslagerungsdatei	5.80 GB
Auslagerungsdatei	C:\pagefile.sys

10.2 Testfälle

Es wurden nur Testfälle durchgeführt, die direkt oder indirekt mit der Erweiterung von IMSES zu tun haben.

10.2.1 Funktionalitätstests

Test ID: 01	
Name	IMSES ausführen
Testvoraussetzungen	-
Testablauf	1. Matlab R2011b starten 2. Mithilfe des Dateibrowsers zum Programmpfad navigieren 3. Auf IMSES.m Rechtsklick -> Run
Erwartetes Resultat:	- IMSES startet unmittelbar nach dem Ausführen - Es gibt keine Fehlermeldungen - alle Bedienelemente werden geladen - Import Tab ist aktiv & ausgewählt

Test ID: 02	
Name	Import File öffnen
Testvoraussetzungen	IMSES wurde gestartet
Testablauf	1. Im Bereich Import auf Button Browse klicken 2. Mithilfe des Dateibrowsers zum Importfile navigieren 3. File auswählen und auf öffnen klicken
Erwartetes Resultat	- Import File wurde geladen - Dateipfad des Import Files wird im Import Bereich angezeigt - Keine Fehlermeldungen

Test ID: 03	
Name	Import Starten
Testvoraussetzungen	- IMSES wurde gestartet - Importfile wurde ausgewählt
Testablauf	1. Auf Start klicken 2. Auf Ende des Importvorgangs warten
Erwartetes Resultat	- Import File wird importiert - Alle Objekte werden eingelesen - Neue Objekte werden erzeugt falls notwendig - Neue Area wird erzeugt falls notwendig - Fehlermeldungen werden ausgegeben falls notwendig - Bestätigung der Vollständigkeit des Imports wird ausgegeben

Test ID: 04	
Name	Simulation durchführen
Testvoraussetzungen	- Import mit Testaplikation „TestApp“ wurde durchgeführt - Fehlendes Child-Object Factor2 wurde erstellt
Testablauf	1. Dem Objekt Factor1 den Wert 2 zuweisen 2. Dem Objekt Factor2 den Wert 5 zuweisen 3. Simulation starten
Erwartetes Resultat	Factor1 muss mit Factor2 multipliziert werden. Das Ergebnis (10) muss in das Objekt Product geschrieben werden.

10.2.2 White Box-Tests

Der Begriff White-Box-Test (auch Glass-Box-Test) bezeichnet eine Methode des Software-Tests, bei der die Tests mit Kenntnissen über die innere Funktionsweise des zu testenden Systems entwickelt werden.

Quelle: http://de.wikipedia.org/wiki/White-Box-Test

Test ID: 05	
Name	Fall 1
Beschreibung	Um den Fall 1 zu testen, muss das Importfile manipuliert werden, damit eine Inkonsistenz gewährleistet ist. Das ist nötig, um eine reale Fehlersituation des Falles 1 zu simulieren. Die Kriterien zum Fall 1 kurz zusammengefasst: - Objekt besitzt ein valider Verweis (in subordinate_list) auf ein Child-Objekt - Das Child-Objekt, auf das verwiesen wird o ist nicht in der ba Datei vorhanden o ist nicht in der XML Chart Datei vorhanden Somit muss ein zufälliges Objekt, auf das verwiesen wird, aus der ba Datei und aus der Chart Datei gelöscht werden.
Testvoraussetzungen	- Import File wurde gemäss Beschreibung angepasst - Import wurde vollständig durchgeführt
Testablauf	1. Auf Start klicken 2. Auf Ende des Importvorgangs warten 3. Objekt Typ des fehlenden Child-Objektes manuell aus der Verweis-ID berechnen 4. Existenz und Korrektheit des fehlenden Child-Objektes im Simulink Plan überprüfen
Erwartetes Resultat	- Das Child-Objekt, auf das verwiesen wird, muss im Simulink Plan (.mdl Datei) vorhanden sein. - Der Objekt Typ des Simulink Objektes muss korrekt sein - Eine Fehlermeldung muss im GUI erscheinen

Test ID: 06	
Name	Fall 2a
Beschreibung	Um den Fall 2a zu testen, muss das Importfile manipuliert werden, damit eine Inkonsistenz gewährleistet ist. Das ist nötig, um eine reale Fehlersituation des Falles 2a zu simulieren. Die Kriterien zum Fall 2a kurz zusammengefasst: - Objekt besitzt ein valider Platzhalter (in subordinate_annotations) eines Verweises auf ein Child-Objekt - Der zugehörige Verweis des Platzhalters ist undefiniert (Verweis-ID = 4194303) oder nicht vorhanden - Das Child-Objekt, auf das verwiesen wird, besitzt den Status „**Connected**“ - Das Child-Objekt ist in der XML Chart Datei vorhanden Es muss sichergestellt werden, dass die Kriterien von Fall 2a im Importfile auftreten.
Testvoraussetzungen	- Import File wurde gemäss Beschreibung angepasst - Import wurde vollständig durchgeführt
Testablauf	1. Auf Start klicken 2. Auf Ende des Importvorgangs warten 3. Im Regelwerk der zu verwendende Objekttyp des Child-Objektes auslesen 4. Existenz und Korrektheit des fehlenden Child-Objektes im Simulink Plan überprüfen
Erwartetes Resultat	- Das Child-Objekt, auf das verwiesen wird, muss im Simulink Plan (.mdl Datei) vorhanden sein. - Der Objekt Typ des Simulink Objektes muss korrekt sein - Das neu erstellte Objekt muss in einer neuen Area abgelegt sein

Test ID: 07	
Name	Fall 2b
Beschreibung	Um den Fall 2b zu testen, muss das Importfile manipuliert werden, damit eine Inkonsistenz gewährleistet ist. Das ist nötig, um eine reale Fehlersituation des Falles 2b zu simulieren. Die Kriterien zum Fall 2b kurz zusammengefasst: - Objekt besitzt ein valider Platzhalter (in subordinate_annotations) eines Verweises auf ein Child-Objekt - Der zugehörige Verweis des Platzhalters ist undefiniert (Verweis-ID = 4194303) oder nicht vorhanden - Das Child-Objekt, auf das verwiesen wird, besitzt den Status „**Owned**“ - Das Child-Objekt ist in der XML Chart Datei vorhanden Es muss sichergestellt werden, dass die Kriterien von Fall 2a im Importfile auftreten.
Testvoraussetzungen	- Import File wurde gemäss Beschreibung angepasst - Import wurde vollständig durchgeführt
Testablauf	1. Auf Start klicken 2. Auf Ende des Importvorgangs warten 3. Im Regelwerk der zu verwendende Objekttyp des Child-Objektes auslesen 4. Existenz und Korrektheit des fehlenden Child-Objektes im Simulink Plan überprüfen
Erwartetes Resultat	- Das Child-Objekt, auf das verwiesen wird, muss im Simulink Plan (.mdl Datei) vorhanden sein. - Der Objekt Typ des Simulink Objektes muss korrekt sein - Das neu erstellte Objekt darf **nicht** in einer neuen Area abgelegt sein, sondern muss in der importierten Area abgelegt sein - Eine Fehlermeldung muss im GUI erscheinen

Test ID: 08	
Name	Fall 3
Beschreibung	Um den Fall 3 zu testen, muss das Importfile manipuliert werden, damit eine Inkonsistenz gewährleistet ist. Das ist nötig, um eine reale Fehlersituation des Falles 3 zu simulieren. Die Kriterien zum Fall 3 kurz zusammengefasst: - Objekt besitzt ein valider Platzhalter (in subordinate_annotations) eines Verweises auf ein Child-Objekt - Der zugehörige Verweis des Platzhalters ist undefiniert (Verweis-ID = 4194303) oder nicht vorhanden - Das Child-Objekt ist **nicht** in der XML Chart Datei vorhanden - Das Child-Objekt, auf das verwiesen wird, besitzt den Status „**Owned**“ Es muss sichergestellt werden, dass die Kriterien von Fall 3 im Importfile auftreten.
Testvoraussetzungen	- Import File wurde gemäss Beschreibung angepasst - Import wurde vollständig durchgeführt
Testablauf	1. Auf Start klicken 2. Auf Ende des Importvorgangs warten 3. Existenz und des fehlenden Child-Objektes im Simulink Plan überprüfen
Erwartetes Resultat	- Das Child-Objekt, auf das verwiesen wird, darf **nicht** im Simulink Plan (.mdl Datei) vorhanden sein. - Eine Fehlermeldung muss im GUI erscheinen

Test ID: 09	
Name	Erzeugen einer eindeutigen ObjectID
Beschreibung	Bei der Erstellung eines neuen Objektes muss dem Objekt eine eindeutige ObjectID zugeteilt werden.
Testvoraussetzungen	- Import wurde vollständig durchgeführt - Fall 1,2 oder 3 ist aufgetreten - Ein neues Objekt wurde erstellt
Testablauf	1. Doppelklick auf das neu erstellte Objekt 2. Im Anzeigefeld ObjectID die ID auf Eindeutigkeit überprüfen
Erwartetes Resultat	- Object ID muss eindeutig sein - Die ID muss nach folgendem Muster gestaltet sein: 11****

Test ID: 10	
Name	Einmaliges Erstellen einer Area
Beschreibung	Wenn ein neues Child-Objekt erstellt wird, das „Connected“ ist, wird eine neue Area erstellt. Wenn zu einem späteren Zeitpunkt nochmals ein „Connected“ Child-Objekt erstellt wird, darf keine neue Area mehr erstellt werden.
Testvoraussetzungen	- Import wurde vollständig durchgeführt - Mehrere neue „Connected“ Objekte wurden erstellt
Testablauf	1. Simulink Objektmodell öffnen 2. Überprüfen, ob mehrere Area erstellt wurden
Erwartetes Resultat	- Nur eine Area wurde erstellt

10.2.3 Testsummary

Test ID: 01	
Testdatum	24.04.2014
Tester	Simon Marty
Erwartetes Resultat	IMSES startet unmittelbar nach dem Ausführen. Es gibt keine Fehlermeldungen und das GUI wird vollständig geladen. Der Tab Import ist ausgewählt.
Testergebnis	IMSES wird wie erwartet gestartet
Bemerkung	-
Status	Bestanden

Test ID: 02	
Testdatum	24.04.2014
Tester	Simon Marty
Erwartetes Resultat	Die Import Datei wird vollständig geladen. Der Dateipfad der Import Datei wird im Import Bereich angezeigt. Es dürfen keine Fehlermeldungen erscheinen.
Testergebnis	Die Import Datei wird wie erwartet vollständig geladen
Bemerkung	-
Status	Bestanden

Test ID: 03	
Testdatum	24.04.2014
Tester	Simon Marty
Erwartetes Resultat	Das Import File wird vollständig importiert und alle Objekte werden eingelesen. Neue zu erstellende Objekte werden erzeugt. Falls notwendig wird eine Neue Area erzeugt. Fehlermeldungen werden ausgegeben, falls notwendig. Eine Bestätigung der Vollständigkeit des Imports wird ausgegeben.
Testergebnis	Der Import wird vollständig durchgeführt, Bestätigungsmeldung erscheint.
Bemerkung	-
Status	Bestanden

Test ID: 04	
Testdatum	24.04.2014
Tester	Simon Marty
Erwartetes Resultat	Factor1 muss mit Factor2 multipliziert werden. Das Ergebnis (10) muss in das Objekt Product geschrieben werden.
Testergebnis	Die Simulation funktioniert. Die Zahlen werden korrekt multipliziert und in dasObjekt Product reingeschrieben.
Bemerkung	Nur mit Applikation „TestApp“ möglich
Status	Bestanden

Test ID: 05	
Testdatum	24.04.2014
Tester	Simon Marty
Erwartetes Resultat	Das Child-Objekt, auf das verwiesen wird, muss im Simulink Plan (.mdl Datei) vorhanden sein. Der Objekt Typ des Simulink Objektes muss korrekt sein. Eine Fehlermeldung muss im GUI erscheinen.
Testergebnis	Das Child Objekt wird mit korrektem Typ erstellt, die Fehlermeldung erscheint.
Bemerkung	-
Status	Bestanden

Test ID: 06	
Testdatum	24.04.2014
Tester	Simon Marty
Erwartetes Resultat	Das Child-Objekt, auf das verwiesen wird, muss im Simulink Plan (.mdl Datei) vorhanden sein. Der Objekt Typ des Simulink Objektes muss korrekt sein Das neu erstellte Objekt muss in einer neuen Area abgelegt sein
Testergebnis	Eine neue Area wird erstellt und das neu generierte Child-Objekt wird darin abgelegt.
Bemerkung	-
Status	Bestanden

Test ID: 07	
Testdatum	24.04.2014
Tester	Simon Marty
Erwartetes Resultat	Das Child-Objekt, auf das verwiesen wird, muss im Simulink Plan (.mdl Datei) vorhanden sein. Der Objekt Typ des Simulink Objektes muss korrekt sein. Das neu erstellte Objekt darf **nicht** in einer neuen Area abgelegt sein, sondern muss in der importierten Area abgelegt sein. Eine Fehlermeldung muss im GUI erscheinen
Testergebnis	Das Child Objekt wird in der importierten Area erstellt und die Fehlermeldung wird ausgegeben
Bemerkung	-
Status	Bestanden

Test ID: 08	
Testdatum	24.04.2014
Tester	Simon Marty
Erwartetes Resultat	Das Child-Objekt, auf das verwiesen wird, darf **nicht** im Simulink Plan (.mdl Datei) vorhanden sein. Eine Fehlermeldung muss im GUI erscheinen.
Testergebnis	Child-Objekt wird nicht erstellt und die Fehlermeldung erscheint.
Bemerkung	-
Status	Bestanden

Test ID: 09	
Testdatum	24.04.2014
Tester	Simon Marty
Erwartetes Resultat	Die Object ID muss eindeutig sein und nach folgendem Muster gestaltet sein: 11****
Testergebnis	Object ID ist eindeutig und nach vorgegebenem Muster aufgebaut
Bemerkung	-
Status	Bestanden

Test ID: 10	
Testdatum	24.04.2014
Tester	Simon Marty
Erwartetes Resultat	Es darf nur eine Area erstellt werden
Testergebnis	Auch bei mehreren neu erstellten „Connected“ Objekten wird nur eine Area erstellt
Bemerkung	-
Status	Bestanden

11. Schlusswort

Die IPA ist nach meiner Meinung sehr gut Verlaufen. Es gab keine grösseren Probleme und alle Funktionalitäten konnten umgesetzt werden. Das Endprodukt entspricht weitgehend den Erwartungen. Meine IPA zeichnete sich vor allem durch umfangreiche Analyse und Planung ab. Das war der anspruchsvollste Teil der IPA.

Wie bei jeder Arbeit gab es auch hier erfreuliche und weniger erfreuliche Momente.

Die Höhen waren:

- Die Implementation konnte relativ schnell und ohne grosse Probleme umgesetzt werden.
- Der Soll-Ist-Zeitplan ist schlussendlich aufgegangen.
- Alle Tests wurden bestanden.

Die Tiefen waren:

- Ständiger Zeitdruck – zu wenig Zeit für detaillierte Auseinandersetzung gewisser Aspekte.
- Unterschätzung des Aufwandes für die technische Dokumentation, vor allem im Analyse Teil

Schlussendlich gab es doch mehr positive als negative Momente. Das fertige Tool kann nun produktiv verwendet werden und alles in allem hat mir die Umsetzung der IPA Spass gemacht.

12. Anhang

Im Anhang befinden sich alle Code-Ausschnitte, welche im Umfang der IPA erstellt wurden. Da nur Code Anpassung in einer Klasse vorgenommen wurde, wird der gesamte Code aufgeführt, und die selbsterstellten Zeilen markiert. Einige Funktionen wurden hier eingerückt, da es sonst Platzprobleme gegeben hätte.

12.1 BA_OBJ.m

```
classdef BA_OBJ < hgsetget
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   (C) Copyright by Siemens Schweiz AG, Building Technologies Group,
%       HVAC Products, 2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Project                     : IMSES
%   Target Hardware             : PC
%   Target Operating System     : WinXP Console
%   Language/Compiler           : Matlab 2010 and higher
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Workfile                    : BA_OBJ.m
%
%   Author                      : Simon Marty
%   Version                     : v1.1
%   Date                        : 20-Feb-2012
%
%   Author                      : Thomas Rohr
%   Version                     : v1.0
%   Date                        : 17-Apr-2014
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Matlab Informations
%   zugriff auf properties von außerhalb nur mit
%   superclass: < hgsetget   ???
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function/Interface:
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Revision History
%   (Put meaningful comments in SourceSafe for log below!)
%   (Please remove blank lines and very old comments!)
%
%   2014-04-17 17:30 Simon Marty
%   Create Object if Object doesn't exist, but has a valid Link ID
%
%   2014-04-16 17:50 Simon Marty
%   Create new Area if a new connected Object is createt
%
%   2014-04-16 14:50 Simon Marty
%   Create Object if the Link ID in the subordinate_list is wrong but a
%   matching Placeholder in subordinate_annotations does exist
%
%   2013-11-07 11:30 Stefan Boetschi
%   Write error message to GUI if property Group-Category or Group-Number
%   is missing in the *.ba file to be imported.
```

```
%
%   2013-10-10 12:00 Stefan Boetschi
%   Support for Group-Master-Objects under construction
%   Support for Group-Member-Objects under construction
%
%   2012-03-20 14:00 Thomas Rohr
%   Header comment was attached
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


properties (GetAccess = public)
  SubList
  ObjectList
  ObjectListPos
  DevIDRef
  ObjIDRef
  ObjectName
  ObjectType
  ObjectNode
  DocNode
  h
  FWLib
  FilePath
  % Array für Regelwerk
  BAObjTypeNames
end
methods(Static)
   function xmlChartName = getXmlChartName(obj)
       % liest den Chart File Name aus und gibt ihn als Funktionswert zurück
       persistent ChartName;
       if isempty(ChartName)
           % XML File vom Full Import Zip File
           xmlChart = dir([obj.FilePath '\*.xml']);
           ChartName = xmlChart.name;
       end
       xmlChartName = ChartName;
   end
   %%
   function ObjID = getUniqueObjID
       % eindeutige ID erzeugen
       % wird als ObjID für neu erzeugte Objekte gebraucht
       persistent ID;
       if isempty(ID)
           ID=11000;
       end
       ID=ID+1;
       ObjID = int2str(ID);
   end
end
```

```
function parse(obj)
    import javax.xml.xpath.*
    factory = XPathFactory.newInstance;
    xpath = factory.newXPath;
    % Initialize BA Obj Types
    obj.BAObjTypeNames = {'R_A' 2;'R_B' 5;'R_M' 19;'R_UNSG' 48;...
                          'R_LGTCMD' 260;'R_BLSCMD' 258;'R_AX' 2;...
                          'R_BX' 5;'R_MX' 19;'R_LGTX' 262;'R_BLSX' 259;...
```

```
                            'W_A' 2;'W_B' 5;'W_M' 19;'CMD_A' 2;'CMD_B' 5;...
                            'CMD_M' 19;'CMD_LGT' 260;'CMD_BLS' 258;'CMD_AX' 2;
                            'CMD_BX' 5};
...
```

Ab hier keine eigene Ergänzung mehr bis zur Ende der Funktion

```
...
end
```

```
function getSubList(obj)
```

In dieser Funktion wurden keine Änderungen vorgenommen…

```
end
```

```
function generateObjectList(obj)
import javax.xml.xpath.*
factory = XPathFactory.newInstance;
xpath = factory.newXPath;
% XML Object vom Chart File
xDoc = xmlread(obj.getXmlChartName(obj));
% Wenn NewArea=1, wird eine neue Area für alle neuen connected Objekte
  erstellt
NewArea=0;
% Wenn Create=1, wird das Objekt erzeugt, ansonsten nicht
Create=1;
if isempty(obj.SubList)
    Send2GUI('    ERROR : Es wurde kein AF Object gefunden',obj.h);
else
   count = 1;
   %die Sublist wird durchsucht
   for k=1:size(obj.SubList,2)
        exprXML=xpath.compile(['.//FB[@Name="' obj.SubList{3,k} '"]/@Typ']);
        % Chart File nach Sublist Objektnamen durchsuchen
        resultXML=exprXML.evaluate(xDoc, XPathConstants.STRING);
        exprBA=xpath.compile(['.//EOParameter[@Name="Object_Identifier"]...
                              //Value[@Value="' obj.SubList{2,k} '"]']);
        resultBA=exprBA.evaluate(obj.DocNode, XPathConstants.NODE);
        %[EngineeringObject]

        if ~strcmpi(obj.SubList{3,k},'') && ~strcmpi(obj.SubList{3,k},'~')
        ...&& strcmpi(obj.SubList{2,k},'4194303')
            % Subordinate_Annotations enthält Platzhalter für ein
            % Objekt, welches die Subordinate_List NICHT enthält.
            % Objekt ist in XML Chart vorhanden
            % -> Neues Objekt erzeugen
            % Fall 2

            if strcmpi(obj.SubList{4,k},'false')
                % für SubList Elemente = Connected
                % Neue Area für alle connected Elemente anlegen
                obj.ObjectList{count}=BA_VN_F;
                % Objekt = ViewNode Function
                obj.ObjectList{count}.ObjectType='BA_VN_F';
                % Platzierungs Ort für neue Area ermitteln
                SubString=obj.ObjectName;
                slash=strfind(SubString,'/');
                TestName=SubString(1:slash);
                TestName=[TestName 'BA/ConnectedArea'];
                obj.ObjectList{count}.ObjectName=TestName;
```

```
                % count erhöhen, nächstes Objekt
                count = count + 1;
                NewArea=1;

            else
                % owned, eine neue Area muss nicht erstellt werden
                NewArea=0;
            end
            if ~isempty(resultXML)
                % wenn Object Type im Chart gefunden wurde
                for c=1:size(obj.BAObjTypeNames)
                    if strcmpi(obj.BAObjTypeNames(c,1),resultXML)
                        % Object type Nummer von Regelwerk herauslesen
                        ProtoType=obj.BAObjTypeNames{c,2};
                    end
                end
                if isempty(ProtoType) || ProtoType==0
                    % Wenn nichts im Chart File gefunden, kein Objekt
                      erzeugen
                    Create=0;
                    Send2GUI({...
                    ['    ERROR : Object Type of Object ' obj.SubList{3,k} '
                                  could not be defined'];...
                     '            The Object is not going to be created'}
                      ,obj.h);
                else
                    % Objekt Type konnte ermittelt werden, Objekt wird
                      erzeugt
                    Create=1;
                end
            else
                % Subordinate_Annotations enthält Platzhalter für ein
                % Objekt, welches die Subordinate_List NICHT enthält.
                % Objekt ist in XML Chart NICHT vorhanden
                % Fall3
                if strcmpi(obj.SubList{4,k},'true')
                    Send2GUI({...
                    ['    ERROR : Sublist Object ' obj.SubList{3,k} ' has no
                          valid Object ID'];...
                   '} ,obj.h);
                end
                % Kein Objekt erzeugen
                Create=0;
            end
        elseif isempty(resultBA)
            % Fall 1
            % Verweis ist in Subordinate_List vorhanden. Das
            % Objekt, auf das verwiesen wird, ist jedoch nicht
            % vorhenden -> es muss erstellt werden

            Send2GUI({...
              '    ERROR : Engineering Object not found. This could be an
                           Error in the *.ba Export File.' ;...
                           Subordinate List of Object: | ' obj.ObjectName ' |
                           at Position ' num2str(k)   ];...
                           BaObjRef.DeviceId : ' obj.SubList{1,k}
                           BaObjRef.ObjectId : ' obj.SubList{2,k}
                           Object Name       : ' obj.SubList{3,k}
                           Valid for Import  : ' obj.SubList{4,k}
            ]},obj.h);
```

```
            ProtoType = str2double (obj.SubList{2,k});
            ProtoType = bitand(ProtoType, hex2dec('FFC00000'));
            ProtoType = bitshift(ProtoType, -22);
            Create=1;
        else
            % Normalfall
            if strcmpi(obj.SubList{4,k},'true')
                % nur owned Objekte erstellen
                ObjectNode_v =
            resultBA.getParentNode.getParentNode.getParentNode.getParentNode;
                exprBA=xpath.compile('.//EOTypeRef//@Prototype');
                ProtoType = exprBA.evaluate(ObjectNode_v,
                                            XPathConstants.NUMBER);
                Create=1;
            else
                Create=0;
            end
        end
        if Create == 1
            %Hier jetzt mit Switch Case Bewerten und Richtiges BACNET Objekt
            %zuweisen.
            % 1. in Liste eintragen Objekt{1,:}
            switch ProtoType
                case{1,2,3,4,5,6,11,12,13,14,17,32}
                obj.ObjectList{count}=BA_SIMPEL;
                switch ProtoType
                    case {1} % SIE_BA_ANALOGINPUT = 1, // AI
                        obj.ObjectList{count}.ObjectType='BA_AI';
                    case {2} % SIE_BA_ANALOGPROCESSVALUE = 2 // APrcVal
                        obj.ObjectList{count}.ObjectType='BA_AV';
                    case {3} % SIE_BA_ANALOGOUTPUT = 3, // AO
                        obj.ObjectList{count}.ObjectType='BA_AO';
                    case {11} % SIE_BA_ANALOGCONFIGVALUE = 11 // ACnfVal
                        obj.ObjectList{count}.ObjectType='BA_ACNV';
                    case {12} % SIE_BA_ANALOGCALCULATEDVALUE = 12 // ACalcVal
                        obj.ObjectList{count}.ObjectType='BA_ACV';
                    case {4} % SIE_BA_BINARYINPUT = 4, // BI
                        obj.ObjectList{count}.ObjectType='BA_BI';
                    case {5} % SIE_BA_BINARYPROCESSVALUE = 5 // BPrcVal
                        obj.ObjectList{count}.ObjectType='BA_BV';
                    case {6} % SIE_BA_BINARYOUTPUT = 6, // BO
                        obj.ObjectList{count}.ObjectType='BA_BO';
                    case {13} % SIE_BA_BINARYCONFIGVALUE      = 13 // BCnfVal
                        obj.ObjectList{count}.ObjectType='BA_BCNV';
                    case {14} % SIE_BA_BINARYCALCULATEDVALUE = 14 // BCalcVal
                        obj.ObjectList{count}.ObjectType='BA_BCV';
                    case {17} % SIE_BA_UNSIGNEDCONFIGVALUE = 17 // UCnfVal
                        obj.ObjectList{count}.ObjectType='BA_UNSGCNV';
                    case {32} % SIE_BA_COMMAND = 32 // CmdObj
                        obj.ObjectList{count}.ObjectType='BA_CMD';
                end

                obj.ObjectList{count}.
                ObjectName=strcat(obj.ObjectName,'/',obj.SubList{3,k});

                case{7,8,9,15,16}
                obj.ObjectList{count}=BA_MULTI;
                switch ProtoType
                    case {7} % SIE_BA_MULTISTATEINPUT = 7, // MI
                        obj.ObjectList{count}.ObjectType='BA_MI';
```

```
        case {8} % SIE_BA_MULTISTATEPROCESSVALUE = 8 // MPrcVal
            obj.ObjectList{count}.ObjectType='BA_MV';
        case {9} % SIE_BA_MULTISTATEOUTPUT = 9, // MO
            obj.ObjectList{count}.ObjectType='BA_MO';
        case {15} % SIE_BA_MULTISTATECONFIGVALUE  = 15 // MCnfVal
            obj.ObjectList{count}.ObjectType='BA_MCNV';
        case {16} % SIE_BA_MULTISTATECALCULATEDVALUE = 16 //
                    MCalcVal
            obj.ObjectList{count}.ObjectType='BA_MCV';
    end
    obj.ObjectList{count}.ObjectName=
    strcat(obj.ObjectName,'/',obj.SubList{3,k});

    %% Stefan, 04.10.13: UNDER CONSTRUCTION %%
    case {29} % SIE_BA_GROUPMASTER = 29 // GrpMaster
    Send2GUI('    WARNING : Support for BA_GROUP_MASTER
            (GrpMaster) is currently under construction',obj.h);
    obj.ObjectList{count}=BA_GROUPMASTER;
    obj.ObjectList{count}.ObjectType = 'BA_GROUP_MASTER';
    obj.ObjectList{count}.ObjectName = strcat(obj.ObjectName,'/',
    obj.SubList{3,k});

    % comment the continue statement in order to make the
    % BA_GROUP_MASTER objects appear in the ObjectList:
    % continue % Jump to next Iteration
    %% Stefan, 10.10.13: UNDER CONSTRUCTION %%
    case {30} % SIE_BA_GROUPMEMBER = 30 // GrpMbr
    Send2GUI('    WARNING : Support for BA_GROUP_MEMBER (GrpMbr)
    is currently under construction',obj.h);
    obj.ObjectList{count}=BA_GROUPMEMBER;
    obj.ObjectList{count}.ObjectType = 'BA_GROUP_MEMBER';
    obj.ObjectList{count}.ObjectName = strcat(obj.ObjectName,'/',
    obj.SubList{3,k});

    % OBJ_STRUCTURED_VIEW
    case {40} % SIE_BA_VNOBJECTFUNCTIONAL = 40 // FnctView
    obj.ObjectList{count}=BA_VN_F;
    obj.ObjectList{count}.ObjectType='BA_VN_F';
    obj.ObjectList{count}.ObjectName=strcat(obj.ObjectName,'/',
    obj.SubList{3,k});

    case {41} % SIE_BA_VNOBJECTCOLLECTION = 41 // ColView
    obj.ObjectList{count}=BA_VN_C;
    obj.ObjectList{count}.ObjectType='BA_VN_C';
    obj.ObjectList{count}.ObjectName=
    strcat(obj.ObjectName,'/',obj.SubList{3,k});

    case {44} % SIE_BA_VNOBJECTAREA = 44 // AreaView
    Send2GUI({...
        '    ERROR : There should be only on AreaView Object
        (ProtoType 44) in the *.ba File';...
        '            Proof if there is an Error in the *.ba
         File'} ,obj.h);
    continue % Jump to next Iteration
```

```
                otherwise
                Send2GUI({...
                    ['    ERROR : ProtoType ' ProtoType ' not
                          identified'];...
                          The ProtoType number is wrong in the *.ba  File or
                          not implemented.';...
                          Implement the ProtoType in function
                          generateObjectList(obj) in BA_OBJ.m';...
                          or write a warning/hint message for it.'} ,obj.h);
                continue % Jump to next Iteration
            end
            %%
            if NewArea==1
                % Bei Erzeugung von fehlenden connected Objekten
                % muss neue Area erstellt werden
                obj.ObjectList{count}.ObjectName=[TestName '/'
                                                  obj.SubList{3,k}];
            end

            if  strcmpi(obj.SubList{2,k},'4194303')
                % Dem neuen Objekt eine identische Objekt ID zuweisen
                obj.ObjectList{count}.ObjIDRef=BA_OBJ.getUniqueObjID;
                % Die Objekt ID in der Objekt Liste eintragen
                obj.SubList{2,k}=obj.ObjectList{count}.ObjIDRef;
            else
                obj.ObjectList{count}.ObjIDRef=obj.SubList{2,k};
            end
            % passing the handle
            obj.ObjectList{count}.h=obj.h;
            % passing the FirmwareLib
            obj.ObjectList{count}.FWLib=obj.FWLib;
            % passing the DocNode
            obj.ObjectList{count}.DocNode=obj.DocNode;
            % passing the ObjectNode
            obj.ObjectList{count}.ObjectNode=ObjectNode_v;
            % passing the ObjectList Position
            obj.ObjectList{count}.ObjectListPos=count;
            %aus Sublist mitgeben
            obj.ObjectList{count}.DevIDRef=obj.SubList{1,k};

            if not(isempty(obj.ObjectList{count}))
                obj.ObjectList{count}.parse;
            end

            count=count+1;
        end
    end
end
```