Titel: **Individuelle Projekt Arbeit**

Thema: **Erstellen einer Bedienoberfläche für IMSES (Simulation von Applikationen der Gebäudeautomation)**

Key Words: IPA, IMSES, GUI

Dokument Kategorie:	ProjectRecord
Revision:	1.0
Änderungsdatum:	16.04.2015
Dokument Status:	Fertiggestellt
Autor:	Dominik Zgraggen
Abteilung:	IC BT CPS R&D ZG CS SAP
Verantwortliche Stelle:	dominik.zgraggen@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	30-März-2015	Dominik	Status = Neuerstellung
0.2	31-März-2015	Dominik	Status = In Bearbeitung
0.3	07-April-2015	Dominik	Status = In Bearbeitung
0.4	14-April-2015	Dominik	Status = In Bearbeitung
0.5	15-April-2015	Dominik	Status = In Bearbeitung
1	16-April-2015	Dominik	Status = **Fertiggestellt**

Inhaltsverzeichnis:

Abbildungsverzeichnis:

Tabellenverzeichnis:

1. Einführung

1.1 IPA

IPA steht für individuelle praktische Arbeit und wird von allen Informatik-Lernenden im letzten Semester durchgeführt. Für einen organisatorisch reibungslosen Ablauf der IPA sorgt die Plattform PkOrg.

1.2 Zweck des Dokumentes

Der IPA-Bericht enthält alle Arbeitsschritte, welche im Rahmen der IPA von Dominik Zgraggen durchgeführt wurden. Das Dokument ermöglicht Einsicht in die Überlegungen und Tätigkeiten, die während der IPA gemacht wurden.

1.3 Zielpublikum

Der Inhalt richtet sich in erster Linie an die Experten und den Fachvorgesetzen der IPA. Dadurch kann die Arbeit nachvollzogen und beurteilt werden.
Des Weiteren können Mitarbeiter, wie zum Beispiel andere Entwickler von IMSES, diese Dokumentation verwenden, um sich zu informieren.

2. Projektauftrag

Tabelle 2-1: Projektauftrag

Projekttitel	Erstellen einer Bedienoberfläche für IMSES (Simulation von Applikationen der Gebäudeautomation)
Prüfungskandidat	Dominik Zgraggen
Fachvorgesetzter	Michael Speckien
Auftraggeber	Michael Speckien

2.1 Einführung

Für die Gebäudeautomation werden bei SIEMENS frei programmierbare Controller eingesetzt. Die Applikationssoftware dieser Controller wird aus einer Bibliothek zusammengestellt.
Die Applikationssoftware für die Bibliothek wird vor der Auslieferung unter anderem mit dem Tool IMSES auf Basis Matlab / Simulink (R) getestet. Bisher erfordert der Testvorgang eine aufwändige Einarbeitung in das Tool IMSES und den Testablauf. Diese IPA soll eine Workflow-orientierte Bedienoberfläche bereitstellen, welche die Einarbeitungszeit und die Fehlerquote reduziert.

2.2 Aktuelle Situation, Hintergrund

Das Testen der Applikationen für die Bibliothek erfolgt unter anderem mit dem Tool IMSES. IMSES beinhaltet eine komplexe Funktionalität für verschiedene Use Cases. Ein häufig genutzter Einsatzfall soll der Open-Loop-Test mit simuliertem Controller werden. Der Einsatz wird derzeit selten genutzt, da er durch manuelle Befehlseingaben in MATLAB und durch unzureichende Bedienoberflächen erschwert ist.

2.3 Detaillierte Aufgabenstellung

Um die derzeit seltene Nutzung bei den Applikationsentwicklern zu verbessern, sind folgende Erweiterungen vorgesehen:

- Entwicklung einer Bedienoberfläche, die den kompletten Workflow eines Open-Loop-Tests unterstützt.
- Die Bedienoberfläche soll speziell Einsteigern den Einsatz von IMSES erleichtern.
- Der Workflow und die Ergebnisse sollen erkennbar sein.

2.3.1 IMSES Gesamtüberblick

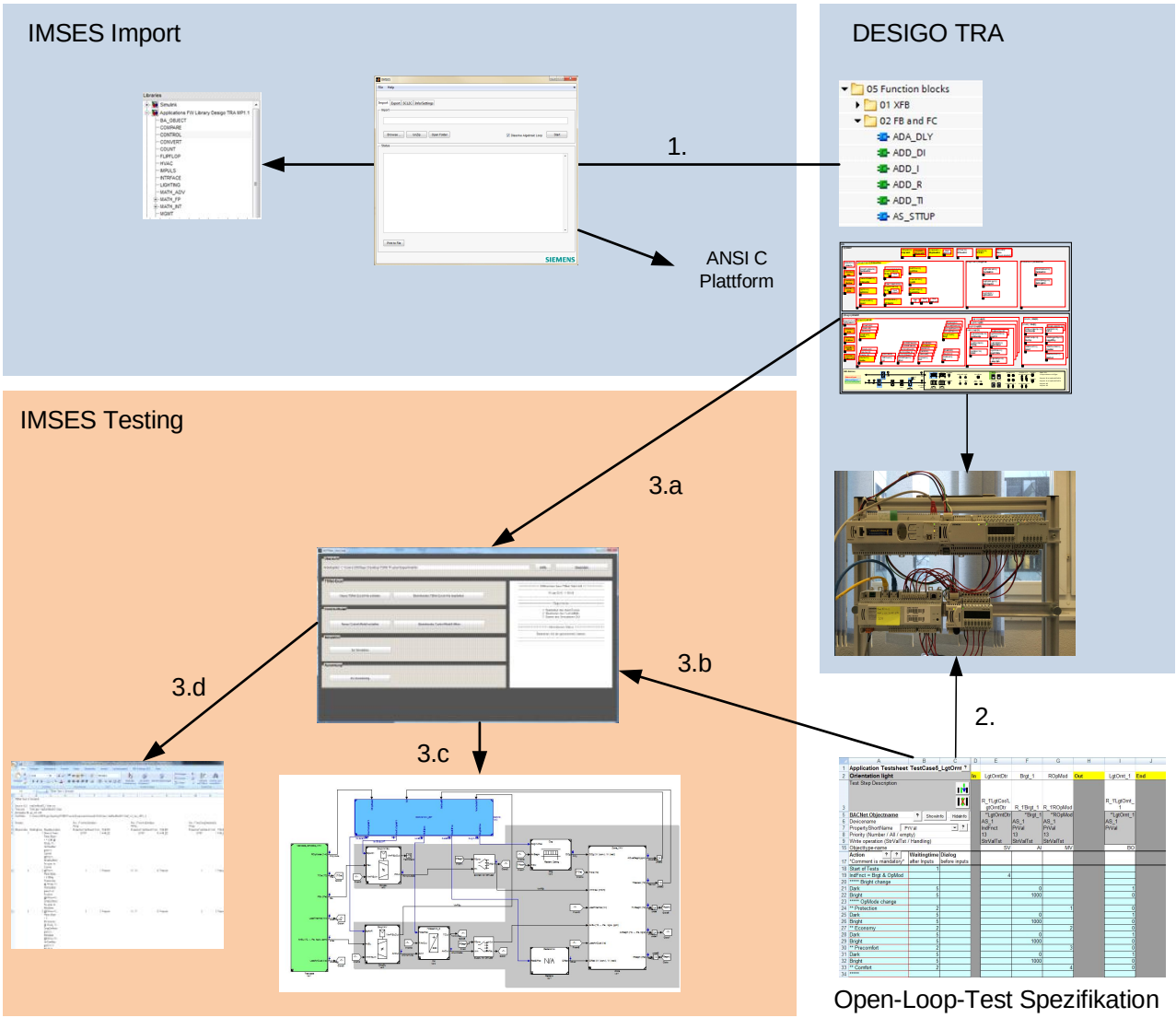


Abbildung 2-1: IMSES Gesamtüberblick

Use Cases:

1. **Import Bausteine**
   Bausteine aus dem TRA-System werden importiert in IMSES, und konvertiert in ANSI C. Danach stehen sie für IMSES zur Verfügung.
   Nutzer: IMSES-Spezialist
   Anwendung: selten

2. **Open-Loop-Test mit Hardware-in-the-Loop** (kein Bestandteil von IMSES)
   Applikationen auf dem Controller werden mit Hilfe einer Testspezifikation getestet (TsNet)
   Nutzer: Applikationsentwickler
   Anwendung sehr häufig

3. **Open-Loop-Test mit IMSES**
   Eine auf TRA entwickelte Applikation wird mit IMSES als simulierter Controller getestet
   a. Import einer Applikation von TRA
   b. Import der Testspezifikation

c. Ablauf Open-Loop-Test
d. Auswertung und Dokumentation des Tests

Nutzer: Applikationsentwickler
Anwendung: sehr häufig

Use Case 3 soll durch die Bedienoberfläche vollständig abgedeckt werden.
Die Funktionen 3.a, b, c und d sind bereits funktionsfähig vorhanden und sind nicht Bestandteil der IPA. Die Funktionen müssen lediglich in die Bedienoberfläche integriert werden.

2.3.2 Voraussetzungen, Vorarbeiten

Im Vorfeld der IPA wurde von Dominik Zgraggen eine Workflow-Analyse durchgeführt. Er erhielt dabei von verschiedenen Mitarbeitern fachkundige Unterstützung. Der gesamte Workflow wurde in einzelne Schritte zerlegt und in einem Flussdiagramm dargestellt. In der dazugehörigen Beschreibung sind für jeden Schritt folgende Gesichtspunkte ausformuliert:

- Name und Inhalt
- Eingangsdaten (externe Daten, IMSES-Interne Daten, Settings, Dateipfade…)
- Eingangsvoraussetzungen (Schritte, die vorher gemacht sein müssen und deren Gesamtergebnis)
- Benutzereingaben (nur wenn zwingend erforderlich)
- Mögliche Stati, Warnungen und Fehler
- Ausgangsdaten (Extern, IMSES-intern)
- Gesamtergebnis (OK / nicht OK / OK mit Warnungen)

Die Ergebnisse der Vorarbeiten befinden sich im Anhang.

2.3.3 Anforderungen

Die Nutzer sind Applikationsentwickler mit geringen IMSES-Kenntnissen.

- Der Nutzer soll entsprechend dem Workflow des Use Cases durch den Test geführt werden.
- Der Benutzer kann einzelne Schritte oder mehrere nacheinander anwählen.
- Der Workflow kann jederzeit vom Nutzer unterbrochen werden und später fortgesetzt werden.
- Einzelne Schritte können jederzeit mit gleichen oder geänderten Eingangsdaten wiederholt werden.
- Der Gesamtstatus über alle Schritte soll vom Nutzer jederzeit erkennbar sein.
- Benutzereingaben sollen, wenn immer möglich, zu Beginn über „Settings“ erfolgen, so dass im Normalfall während des Testings keine weiteren Eingaben erfolgen müssen.
- Die Bedienoberfläche beinhaltet eine Help-Funktion, die den Ablauf und die Eingabefelder erläutert.

2.4 Projektziele

Im Rahmen der IPA soll IMSES um die definierten Erweiterungen ergänzt werden.

Die erstellte Software ist unmittelbar nach der Realisierung zu testen.
Testkonzept: White-Box-Ansatz mit Stichproben, Akzeptanztest bei Applikationsentwickler

2.5 Projektumfang

Das Projekt umfasst folgende Punkte:

1. Konzept einer Bedienoberfläche für den Open-Loop-Test mit IMSES
2. Erstellen des Layouts
3. Implementierung der Bedienoberfläche
4. Einbindung in die bestehende IMSES-Umgebung
5. Testen der Software

Bei Erstellung der Workflowanalyse hat sich gezeigt, dass der Gesamtworkflow komplexer ist als angenommen. Ausserdem wurde festgestellt, dass einige bereits existierende Funktionen nicht unverändert in die Bedienoberfläche integriert werden können.

Daher wird für die IPA vereinbart:

Die Bedienoberfläche enthält alle Elemente des Gesamtworkflows.
Folgende Elemente des Gesamtworkflows werden funktionsfähig implementiert:

- Öffnen eines vorhandenen Projekts
- Auswählen eines Tests
- Durchführen der Simulation

Folgende Elemente werden nur in der Bedienoberfläche vorgesehen, müssen aber nicht zwingend funktionsfähig implementiert werden:

- Projektverwaltung
- Import einer Applikation von TRA
- Teststatus-Auswertung

3. Projektorganisation

3.1 Datensicherung

Die Matlab-Installation ist lokal, daher wird das Projekt mit dem betreffenden Code jeden Abend auf den Siemens-Server kopiert. Auf dem Server befindet sich die IPA-Ordnerstruktur. Von dem Server wird täglich automatisch ein Backup gemacht. Zusätzlich wird vom IPA-Ordner abends eine ZIP-Datei erstellt, welche auf einer lokalen Festplatte abgelegt wird. Dadurch kann jederzeit auf alle bisherigen Versionen zurück gegriffen werden.

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 - HUS Struktogrammer
Verwendete Tools	- Snipping Tool - GUIDE (Bestandteil von MATLAB)

3.3 Verwendete Projektmanagementmethode

Bei der Arbeit an dem Projekt wird nach dem Schema von IPERKA vorgegangen:

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

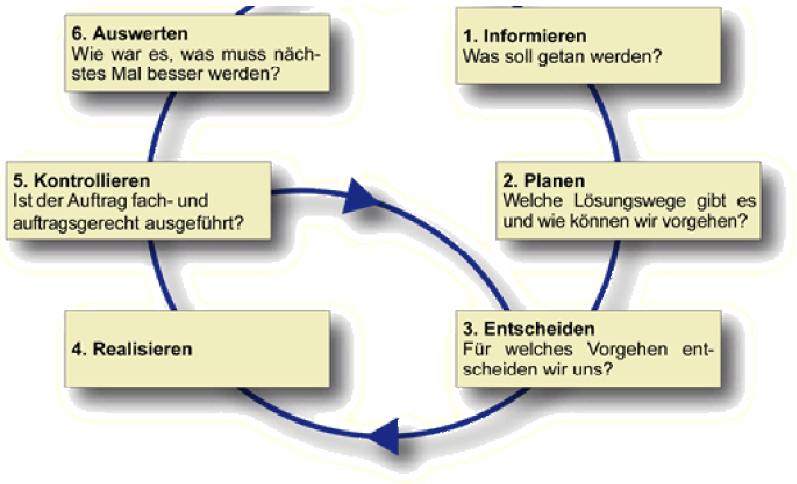


Abbildung 3-1: IPERKA

Begründung der Wahl:
IPERKA passt sehr gut zum Arbeitsstil des IPA-Ausführenden. Ausserdem eignet sich die Unterteilung von Planen und Entscheiden für die IPA. Planen steht für Projekt- und Zeitplanung, Entscheiden für den GUI-Entwurf.

3.4 Ordnerstruktur auf Server

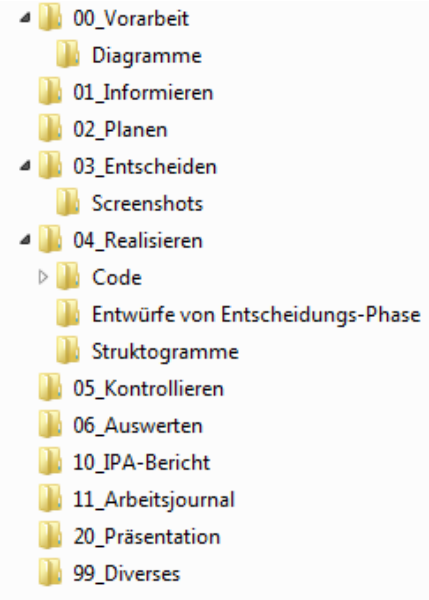


Abbildung 3-2: Ordnerstruktur

3.5 Arbeitsplatz

Der Arbeitsplatz befindet sich in Zug am Zählerweg 5 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. Mögliche Probleme sind zum Beispiel die technische Machbarkeit oder die Abhängigkeit von Dritten.

Matlab-Kenntnisse	Da IPA-Ausführende nicht viel Routine im Umgang mit Matlab besitzt, könnten sich während der Implementierung unvorhergesehene Verzögerungen ergeben. Falls darum der Terminplan nicht eingehalten werden kann, könnte es sein, dass Bestandteile nicht implementiert und getestet werden können. **Wahrscheinlichkeit**: mittel **Auswirkungsgrad**: schwer
Know-How im Umgang mit Testdaten	In der Testphase müssen die Testdaten so manipuliert werden, dass möglichst viele Testfälle damit abgedeckt werden können. Für das nötige Know-How im Umgang mit den Testdaten gilt es eventuell Hilfe anzufordern. **Wahrscheinlichkeit**: mittel **Auswirkungsgrad**: leicht
Akzeptanztest eines Applikationsentwicklers	Mit der ausgewählten Testperson wird vorgängig ein Termin abgemacht. Falls die Person verhindert sein sollte, muss kurzfristig ein Ersatz aufgesucht werden. **Wahrscheinlichkeit**: gering **Auswirkungsgrad**: leicht

Akzeptanztest schlägt fehl	Falls der Akzeptanztest fehlschlagen würde, könnte innerhalb der IPA keine Korrekturen durchgeführt werden. Das Risiko kann durch erfolgreiches Implementieren und durch vorgängige Tests entschärft werden. **Wahrscheinlichkeit**: mittel **Auswirkungsgrad**: schwer

Tabelle 3-2: Risikobeschreibung

3.7 Glossar

BEGRIFF	BESCHREIBUNG
ABT	Automation Building Tool. Tool zum Engineering der Applikationssoftware für das Gebäudeautomationssystem DESIGO TRA
Applikation	Hier: Die gesamte Steuerungs- und Regelungssoftware für einen Gebäudeautomations-Controller
Applikationsfunktion	Teil einer Applikation, bestehend aus Charts und BA Objekten. Beispiel : Raumregelung mit einem Radiator mit Warmwasserventil.
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.
Bausteine	Geschlossene Softwaremodule mit Ein- und Ausgangsschnittstelle, die der Engineer in seiner Software grafisch zu einer Automationslösung verschaltet. Beispiel: PID-Regler
Callback-Funktion	Prozedur im MATLAB, welche von einem GUI-Element aufgerufen wird. (Beispiel: Drücken eines Buttons)
Chart	Plan, auf dem der Engineer mit dem Softwarebausteine platzieren, parametrieren und untereinander verschalten kann. Licht und Jalousien in einem Raum werden grafisch mit Charts Die Regelung- und Steuerungslogik für die Automation der HLKGeräte, und Bausteinen programmiert.
Closed-Loop-Test	Test, bei dem eine Applikation gegen einen simulierten Prozess gefahren wird (Prozess hier: ein Raum mit seinen Komponenten wie Heizkörper oder Fenster). Die Simulation erfasst die Reaktionen der Applikation und gibt abhängig davon die Eingangsbedingungen an die Applikation. Beispiel: Aussentemperatur sinkt à Raumtemperatur sinkt à Heizventil fährt auf à Raumtemperatur steigt wieder à Heizventil fährt zu. Haupteinsatzgebiet: Regelung und Optimierung
Control-Model	Simulink-Model, welches die Applikation enthalt. Enthält Interface für die Testdaten.
DESIGO	Markenname für ein Gebäudeautomationssystem von SIEMENS. Es umfasst Raumautomation, Primäranlagen und Managementstationen
GUI	"Graphical User Interface" Damit ist die graphische Benutzer-/Bedienoberfläche eines Programms gemeint.

Hardware-in-the-loop	Test, bei dem der Controller real vorhanden ist, und über eine Schnittstelle mit der Testumgebung verbunden ist.
HLK	Heizungs-, Lüftungs- und Klimatechnik.
IMSES	Interface Matlab/Simulink Engineering System. Auf Matlab/Simulink basierte Entwicklungs- und Testumgebung für DESIGO TRA.
Info-Box	GUI-Element, welches mit Farben und Status-Meldungen kommuniziert.
IPERKA	Projektmanagement-Methode: **I**nformieren, **P**lanen, **E**ntscheiden, **R**ealisieren, **K**ontrollieren, **A**uswerten
MATLAB	Software/Entwicklungsumgebung von Mathworks Inc zur numerischen Lösung mathematischer Probleme.
Open-Loop-Test	Test, bei dem eine Applikation gegen einen Testaufbau gefahren wird, der die Vorgabe von Eingangsbedingungen und das Beobachten der Reaktionen auf diese ermöglicht. (Beispiel: Raumtemperatur sinkt → Heizventil fährt auf) Haupteinsatzgebiet: Steuerungsaufgaben
Panel	GUI-Element in Matlab, welches eine Art Rahmen um andere GUI-Elemente ziehen lässt.
Report	Auswertung eines Verfahrens (Abschlussübersicht, Ergebnis)
Sheet	= Excel-Tabellenblatt Eine TsNet-Datei beinhaltet mehrere Sheets. beispielsweise Testscripts oder Reports.
Simulierter Controller	Test, bei dem der Controller über eine Software innerhalb der Testumgebung simuliert wird
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
TRA	Raumautomationssystem für HLK, Licht und Beschattung, Bestandteil von DESIGO
TsNet	Testtool zum Open-Loop Test für Applikationen. Es besteht unter anderem aus einem Excel-File zur Testspezifikation mit der Definition von Eingangsbedingungen und den erwarteten Reaktionen. Testscripts ermöglichen das automatisierte Abfahren und Auswerten dieser Testspezifikation.
White Box Test	Test bei dem man Zugriff auf den Code hat
ZIP	Komprimiertes Dateiformat

Tabelle 3-3: Glossar

3.8 Quellen

Tabelle 3-4: Quellenverzeichnis

Nr.	Thema	Quelle	Datum
1	Projektauftrag	https://extranet.pkorg.ch/	30.03.2015
2	IPERKA	http://tgabathuler.ch/IPERKA/Index.html	30.03.2015
3	Matlab Hilfe	http://www.mathworks.ch/matlabcentral/	31.03.2015

[n] = Verweis auf Quelle Nummer

4. Planung

4.1 Zeitplan

IPA Dominik Zgraggen		Abhängigkeit		Aufwand			Status		Geplanter Ablauf																			
Arbeitsschritt	Tätigkeiten	Voraussetzung	Nächster Schritt	Soll [h]	Ist [h]	Abweichung [%]	Meilenstein [Datum]	Status	30.03.2015		31.03.2015		02.04.2015		Karfreitag	Ostermontag	07.04.2015		09.04.2015		10.04.2015		13.04.2015		14.04.2015		15.04.2015		16.04.2015	
									Soll	Ist	Soll	Ist	Soll	Ist			Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist	Soll	Ist
#100	**Informieren**																													
#101	Projektauftrag lesen und Workflow verstehen			0.5	0.5	0		OK	0.5	0.5																				
#200	**Planen**																													
#201	Tätigkeiten und Meilensteine finden und beschreiben		#202	2.0	2.0	0		OK	2.0	2.0																				
#202	Zeitplan (soll) mit Geplantem Ablauf	#201		2.0	2.5	-25	30.03.2015	angep.	2.0	2.5																				
#203	Vorlagen und IPA-Bericht aufbauen			1.0	0.5	50		OK	1.0	0.5																				
#204	Arbeitsumgebung einrichten			0.0	0.0	0		erfolgt																						
#300	**Entscheiden**																													
#301	GUI-Konzept erarbeiten			5.0	5.0	0		OK			5.0	4.5		0.5																
#302	GUI-Entwurf erstellen		#303	5.0	4.5	10		OK					5.0	4.5																
#303	GUI-Entwurf optimieren	#302		2.0	2.0	0		OK					1.0	1.0			1.0	1.0												
#304	Akzeptanztest-Spezifikation ermitteln	#303	#305	3.0	3.8	-26.7		OK									3.0	3.8												
#305	Review GUI-Konzept	#303		0.5	0.3	40	07.04.2015	OK						0.2			0.5	0.1												
#400	**Realisieren**																													
#401	Oberfläche mit GUIDE erstellen inkl. Benennung			1.0	1.0	0		OK											1.0	1.0										
#402	Struktogramme Callback-functions			3.0	3.0	0		Ok											3.0	3.0										
#403	Coding1: Callback-functions implementieren	#401		5.0	7.5	-50		Ok											2.0	3.0	3.0	4.5								
#404	Coding2: Einbinden des bestehenden Codes zur Simulierung			3.0	2.0	33.3		Ok													3.0	2.0								
#405	White-Box Testfälle ermitteln	#403		3.0	2.5	16.7		Ok															3.0	2.5						
#406	Coding3: Usability Improve / Error Handling			2.0	3.0	-50		Ok															2.0	3.0						
#407	Help-Funktion erstellen	#401		2.0	2.8	-40	13.04.2015	Ok															1.0	1.3	1.0	1.5				
#500	**Kontrollieren**	Ok																												
#501	White-Box-Test durchführen	#405		4.0	4.0	0		OK																	2.0	3.0	2.0	1.0		
#502	Akzeptanz-Test durchführen	#303		2.0	2.0	0	15.04.2015	OK																			2.0	2.0		
#600	**Abschliessen**																													
#601	Abgabe: drucken, binden, upload			2.0	3.5	-75	16.04.2015																						2.0	3.5
#700	**Diverses**																													
#701	Zeitplan (ist) einfügen	#202		1.0	1.0	0			0.1	0.1	0.1	0.1	0.1	0.2			0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0,1	0.1	0.1	0.1	0.1
#702	Arbeitsjournal führen	#203		5.0	5.7	-14			0.5	0.7	0.5	0.7	0.5	1.0			0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.3
#703	IPA-Bericht führen	#203		13.0	15.3	-17.7					1.5	2.0					2	2.0							3.5	3.5	2.0	3.5	4.0	4.3
#704	Meeting mit Auftraggeber			2.5	2.7	-8			0.5	0.5			0.5	0.5					0.5	0.5			0.5	0.5			0.5	0.7		
#705	Gespräch mit Erstexpert (Kick-Off)			1.5	1.5	0		OK	1.5	1.5																				
#706	Administratives / Organisatorisches			1.0	1.4	-40			0.1	0.1	0.1	0.1	0.1	0.1			0.1	0.3	0.1	0.0	0.1	0.4	0.1	0.1	0.1	0.0	0.1	0.3	0.1	0.0
#800	**Puffer / Reserve**																													
#801	Pufferzeit			8.0	0.0				0.8	0.0	0.8	0.0	0.8	0.0			0.8	0.0	0.8	0.0	0.8	0.0	0.8	0.0	0.8	0.0	0.8	0.0	0.8	0.0
	Total:			80.0	80.0	0.0			9.0	8.4	8.0	7.4	8.0	8.0			8.0	7.8	8.0	8.1	7.5	7.5	8.0	8.0	8.0	8.5	8.0	8.1	7.5	8.2

x	Geplanter Ablauf (Soll)
x	Positive Abweichung vom Soll
x	Negative Abweichung vom Soll (Verzug)
	Meilenstein

Tabelle 4-1: Zeitplan

4.2 Tätigkeiten und Meilensteine

Nr.	Meilensteine	Arbeitsschritte	Datum
#202	Zeitplanung abgeschlossen (Abgabe an Expert)	#101 bis und mit #202	30.03.2015
#304	Entscheidugnsphase / GUI-Entwurf fertig	#203 bis und mit #305	07.04.2015
#407	Realisierungs-Phase	#401 bis und mit #407	13.04.2015
#502	Kontroll-Phase	#501 und #502	15.04.2015
#601	Abgabe / Projektende	#601	16.04.2015

Tabelle 4-2: Meilensteine

Nr.	Tätigkeit	Beschreibung
#101	Projektauftrag lesen und Workflow verstehen	Detailbeschreibung des Auftrags gemäss PkOrg und die im Vorfeld erstellte Workflow-Analyse verstehen.
#201	Tätigkeiten und Meilensteine finden und beschreiben	Arbeitseinheiten finden und beschreiben, anschliessend Meilensteine
#202	Zeitplan (soll) mit Geplantem Ablauf	Zeitplanung: Soll-Zeit-Spalte für jede Tätigkeit und Geplanter Ablauf festlegen (wann wird welche Tätigkeit ausgeführt)
#203	Vorlagen und IPA-Bericht aufbauen	Vorlagen für Arbeitsjournal und Testspezifikation erstellen und IPA-Bericht strukturieren
#204	Arbeitsumgebung einrichten	Wurde bereits vor IPA eingerichtet (mit der Vorarbeit)
#301	GUI-Konzept erarbeiten	Es wird die Grundlegende Funktionsweise der Benutzeroberfläche erarbeitet: Wie wird der Benutzer durch den Workflow geführt? Fenster, Tabs,…? (ohne auf GUI-Elemente wie Buttons usw. einzugehen)
#302	GUI-Entwurf erstellen	Die einzelnen Workflowschritte werden im GUI dargestellt, dass am Ende das ganze GUI entworfen ist.
#303	GUI-Entwurf optimieren	Das bestehende GUI von #302 wird überarbeitet und nochmals durchdacht.
#304	Akzeptanztest-Spezifikation ermitteln	Es wird mit Hilfe der Anforderungen der User-Akzeptanztest ermittelt, welcher in Schritt #502 von einem Mitarbeiter durchgeführt wird.
#305	Review GUI-Konzept	Das GUI-Konzept wird mit einem Mitarbeiter besprochen und wenn nötig angepasst.
#401	Oberfläche mit GUIDE erstellen inkl. Benennung	Matlab GUIDE ermöglicht das GUI graphisch zusammenzustellen. Die Elemente können auf die Oberfläche gezogen und angepasst werden. Dieser Schritt bildet die Grundlage für die zuerstellende Software.
#402	Struktogramme Callback-functions	Die etwas komplexeren User-Interaktionen auf dem GUI werden mit einem Struktogramm dargestellt.

#403	Coding1: Callback-functions implementieren	Den GUI-Elementen, welche eine Interaktion ermöglichen sollen, wird die Funktion verliehen (gemäss Struktogramm implementiert)
#404	Coding2: Einbinden des bestehenden Codes zur Simulierung	Die Schnittstelle im Fremdcode muss gefunden werden. Der bestehende Code wird sinnvoll integriert und wo nötig angepasst.
#405	White-Box Testfälle ermitteln	Mit dem Stand nach #404 können die White-Box-Testfälle ermittelt und dann in #502 durchgeführt werden.
#406	Coding3: Usability Improve / Error Handling	Das GUI wird mit Code-Ergänzungen benutzerfreundlicher und kann auf ungewünschte Eingaben usw. reagieren.
#407	Help-Funktion erstellen	Help-Funktion wird integriert und der Help-Text verfasst. (ersetzt Benutzeranleitung)
#501	White-Box-Test durchführen	Test durchführen wie er in #405 definiert wurde.
#502	Akzeptanz-Test durchführen	Test wird von einem Mitarbeiter durchgeführt. Als Hilfe dient ihm die in #304 erstellte Testspezifikation.
#601	Abgabe: drucken, binden, upload	IPA-Bericht wird mit Deckblatt 1 gedruckt, gebunden und dann dem Fachvorgesetzten zugestellt. Zweites Exemplar wird dem Zweitexperten zugestellt. Alle Dokumente und Anhänge mussen rechtzeitig auf PkOrg hochgeladen werden.
#701	Zeitplan (ist) einfügen	An Jedem Tag wird der Zeitplan mit der Ist-Stundenanzahl ergänzt. So wird auch erkannt, ob man im Verzug ist.
#702	Arbeitsjournal führen	Das Arbeitsjournal wird an jedem Abend verfasst.
#703	IPA-Bericht führen	Im IPA-Bericht werden alle Arbeitsschritte und Bestandteile des Projekts beschrieben. Das Verfassen und Zusammentragen erfordert viel Aufwand.
#704	Meeting mit Auftraggeber	Geplant ist an jedem zweiten Tag für eine halbe Stunde mit dem Auftraggeber/Fachvorgesetzen die Arbeit zu besprechen.
#705	Gespräch mit Erstexpert (Kick-Off)	Der Experte kommt vorbei. Es werden Informationen zur IPA ausgetauscht.
#706	Administratives / Organisatorisches	z.B. schreiben von Email, vereinbaren von Terminen, Backup machen
#801	Pufferzeit	An jedem Tag 0.8h = insesamt ein Tag Pufferzeit

Tabelle 4-3: Tätigkeiten

5. Arbeitsjournal

5.1 Zweck des Arbeitsjournals

Im Arbeitsjournal werden die täglichen Arbeiten, aufgetretenen Probleme sowie allfällige Hilfestellungen festgehalten. Ausserdem 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

An jedem Tag werden zuerst die gesetzten Ziele genannt, wie sie im Zeitplan aufgeführt sind. Danach folgt für jedes Ziel das Vorgehen und die erreichten Fortschritte. Auch Probleme und getroffene Entscheidungen werden festgehalten. Allfällige Hilfestellungen, der E-Mail Verkehr, die nächsten Schritte und die persönliche Stellungnahme sind ebenfalls Teil des Arbeitsjournals.

5.4 Arbeitsjournale vom 30.03.2014 bis 16.04.2014

Montag, 30.03.2015

Gesetzte Ziele	#101 - Projektauftrag lesen und Workflow verstehen #201 - Tätigkeiten und Meilensteine finden und beschreiben #202 - Zeitplan (soll) mit Geplantem Ablauf #203 - Vorlagen und IPA-Bericht aufbauen
Projektauftrag lesen und Workflow verstehen	Ich habe alle betroffenen Dokumente genau durchgelesen. Den Workflow kannte ich von der Einarbeitungsphase her schon sehr gut.
Tätigkeiten und Meilensteine finden und beschreiben	Ich begann zuerst mit einer Art Brainstorming die Tätigkeiten aufzulisten und entschied mich dazu, jede Tätigkeit in ein bis zwei Sätzen zu beschreiben. Dadurch erhoffte ich mir ein noch konkreteres Bild der Tätigkeit. Die Meilensteine ergaben sich automatisch als ich die Tätigkeiten mit der IPERKA-Methode gliederte.
Zeitplan (soll) mit Geplantem Ablauf	Bei der Darstellung des Zeitplans half mir das Studieren der IPA-Zeitpläne von ehemaligen Lernenden. Beim Schätzen der Soll-Zeit halfen mir die Beschreibungen der Tätigkeiten, trotzdem empfand ich dies als eher schwierig. Vor allem bei den Tätigkeiten im Zusammenhang mit dem GUI-Entwurf hatte ich Mühe den Zeitaufwand zu berechnen. Der geplante Ablauf ergab sich nach der Soll-Zeit-Schätzung praktisch von selbst.
Vorlagen und IPA-Bericht aufbauen	Vorlagen für Arbeitsjournal und Testspezifikationen waren schnell erstellt. Zeitintensiver war jene des IPA-Berichts. Ich nahm eine Dokument-Vorlage der Siemens. Beim Strukturieren irritierte mich ein wenig, dass die Phasen „informieren“ und „planen“ kein eigenes Kapitel im Bericht darstellen werden. Die Aufgabenstellung wurde wie in „PkOrg“ übernommen und im Bericht integriert. Ich musste allerdings noch verschiedene Sätze umschreiben, da die Voranalyse zum jetzigen Zeitpunkt bereits geschehen ist.
Hilfestellungen	Im Meeting mit dem Auftraggeber konnte ich den Zeitplan besprechen und verbessern, bevor ich das Mail an den Erstexperten schickte. Von ihm kam die Idee die acht Stunden Puffer auf alle Tage gleichmässig zu verteilen.
Bemerkung	Zum IPA-Start das Meeting mit dem Erstexperten und Fachvorgesetzten war gut. Ich fühlte mich wohl.
Mailverkehr	Mail mit Excel-Tabelle des Zeitplans an Herrn Peter.
Nächste Schritte	GUI-Entwerfen: Konzept (grundlegender Aufbau und Funktion des GUI)
Fazit	Nach dem Kick-Off Meeting war ich voller Tatendrang und hatte grosse Vorfreude. Trotz kleineren Unsicherheiten beim Erstellen des Zeitplans, bin ich mit dem Verlauf des Tages zufrieden. Der Zeitplan scheint mir im Endeffekt gut gelungen.

Dienstag, 31.03.2015

Gesetzte Ziele	#301 - GUI-Konzept erarbeiten #701 – IPA-Bericht führen
GUI-Konzept erarbeiten	Ich wusste am Anfang nicht wie beginnen und entschied mich die Einleitung zu schreiben. Sie machte mir klar, wie ich Vorgehen möchte. In einem ersten Schritt ermittelte ich drei Möglichkeiten wie das GUI aufgebaut sein könnte. In GUIDE stellte ich die Idee kurz dar. (siehe auch Hilfestellungen) Ich hatte schon früh die Idee in der GUI-Design-Phase eine Nutzwertanalyse zu machen. So entschied ich mich in einem nächsten Schritt Kriterien für den GUI-Aufbau zu beschreiben. Die Möglichkeiten wurden auf die Kriterien untersucht und anschliessend die Nutzwertanalyse durchgeführt. Diese half mir sehr, obwohl ich von ihrer Aussagekraft nicht richtig überzeugt war. Ich zweifelte in erster Linie, weil ich mehr als drei Kriterien erwartet hätte. Anschliessend versuchte ich die Mangelpunkte der ausgewählten Möglichkeit zu beheben oder zumindest zu entschärfen. Ich habe bis jetzt weniger Zeit benötigt als geplant. Allenfalls kann ich am nächsten Tag nach dem Meeting mit dem Auftraggeber, noch Anpassungen vornehmen. Ich habe das Meeting aus diesem Grund für den Morgen angesetzt.
IPA-Bericht führen	Ich habe die Arbeit von #301 im Kapitel „Entscheiden“ eingefügt. Dies erforderte Anpassungen im Inhalts-, Abbildungs- und Tabellenverzeichnis. Ich habe die Kapitel Einführung und Projektorganisation verfasst. Dafür musste ich diverse Projektrisiken erfassen. Ich versuche morgen die Eintrittswahrscheinlichkeit und der Schaden bei Eintritt zu verringern. Ich bin mir ausserdem bewusst geworden, dass ich während der IPA fortlaufend IPA-Kurzfassung, Glossar und Quellen ergänzen muss.
Hilfestellungen	Das Bild für „Möglichkeit 2: Tabs“ wurde aus dem Internet genommen: http://www.mathworks.com/matlabcentral/fileexchange/screenshots/1678/original.jpg (GUIDE unterstütz Tabs nicht)
Bemerkung	Bis jetzt sehr viele Grafiken und Tabellen im IPA-Bericht, stört das?
Mailverkehr	Rückmeldung von Herr Peter zum Zeitplan, Anpassungen durchgeführt.
Nächste Schritte	Meeting mit Auftraggeber um das GUI-Konzept zu besprechen, falls nötig anpassen. Dann GUI-Detail-Entwurf erarbeiten.
Fazit	Ich bin froh, dass es bis jetzt aus meiner Sicht keine grossen Hindernisse gab. Vermutlich ist das GUI-Konzept noch ausbaufähig, ich weiss allerdings nicht wie. Bisher bin ich mit mit meinem IPA-Bericht sehr zufrieden. Ich schätze das Dokumentieren als eine meiner Stärken ein.

Donnerstag, 02.04.2015

Gesetzte Ziele	#302 - GUI-Entwurf erstellen #303 – GUI-Entwurf optimieren
GUI-Konzept erarbeiten	*(UNGEPLANT)* Nach dem Meeting mit dem Auftraggeber machte ich im GUI-Konzept noch eine kleine Verbesserung.
GUI-Entwurf erstellen	Die Workflow-Schritte gewährten viel Spielraum beim GUI-Design. Ich stellte jeden Schritt mehrmals um, bevor ich zufrieden war. Ich war mir nicht sicher, wie genau ich das GUI mit Text im IPA-Bericht beschreiben soll, da es theoretisch selbst erklärend sein sollte. Schliesslich erläutere ich jeden Schritt im GUI in wenigen Sätzen, da ich nicht davon ausgehen kann, dass der Leser zuvor Workflow-Diagramm oder Beschreibung angeschaut hat.
GUI-Entwurf optimieren	Beim Hinzufügen der Buttons „Delete Project“ und „Save Project As“ war ich zuerst etwas verwirrt, dass diese nicht im Workflow vorgesehen waren. Jedoch muss ich eingestehen, dass mit diesem Zusatz eine bessere Benutzerfreundlichkeit erreicht wird.
Hilfestellungen	Meeting mit Auftraggeber: Wir besprachen das Konzept vom Vortag, ein erster Entwurf von heute und wie es mit dem Akzeptanztest aussehen wird.
Bemerkung	Der GUI-Entwurf wurde im Meeting bereits ziemlich genau besprochen, weshalb ich glaube, dass das Review des GUI-Konzepts schon zu einem grossen Teil abgeschlossen ist. Ich konnte ein Teil der Projektrisiken klären.
Mailverkehr	Am Ende wird Herr Peter von mir das Arbeitsjournal zu geschickt bekommen.
Nächste Schritte	GUI weiter optimieren Akzeptanz-Testfälle ermitteln, in Dokument erfassen. Es richtet sich an Tester.
Fazit	Ich wurde etwas nachdenklich, als der Auftraggeber meinte: „Es sieht zwar auf den ersten Blick etwas speziell aus, wird aber gut funktionieren“. Ich kam danach zu verschiedenen Einsichten. Ein GUI designen ist eine Wissenschaft für sich, welche ich vorher in meiner Lehre noch nicht intensiv durchführen konnte. Deswegen wollte ich auch die IPA mit diesem Thema machen. Ich schätze meine Fähigkeiten eine gute Benutzerfreundlichkeit zu erreichen deutlich besser ein, als diejenige ein optisch überzeugendes GUI zu kreieren. Ästhetik war nie meine Stärke. Ich glaube, dass ich diesen Schwachpunkt mit analytischem Denkvermögen und Sachverständnis wett machen kann.

Dienstag, 07.04.2015

Gesetzte Ziele	#303 – GUI-Entwurf optimieren #304 – Akzeptanztest-Spezifikation #305 – Review GUI-Konzept
GUI-Entwurf optimieren	Mir fiel auf, dass das Deaktivieren von Buttons einen sehr guten Einfluss auf die Benutzerführung hat. Ich stellte die Abhängigkeiten der Buttons in einem Flussdiagramm dar und denke, das Diagramm ist aussagekräftig. Jedoch vermute ich, man hätte es mit einer anderen Methode besser darstellen können. Ich wusste allerdings keine.
Akzeptanztest-Spezifikation ermitteln	Die Black-Box-Philosophie stellt für mich keine Probleme dar. Am logischsten war für mich jeweils das erwartete Resultat. Teilweise waren Testvoraussetzungen oder der Testablauf ein wenig verwirrend. Mich machte stutzig, dass der Testablauf eines Testfalls immer sehr kurz war (meistens ein oder zwei Aktionen). Ich wurde mir bewusst, dass ich kurze aber dafür viele Testfälle habe. Im Gedanken an mögliche Fehlerfälle könnte es sein, dass ich ein paar Testfälle vergessen habe. Folgender Fehler, kostete mich etwas Zeit: Ich vergass, dass ich die Funktion einiger Buttons erst nach der IPA implementieren werde und ermittelte fälschlicherweise auch dafür Testfälle.
Review GUI-Konzept	Das „OK“ für meinen GUI-Entwurf von dem Auftraggeber erhielt ich ja schon am Vortag. Heute ging ich kurz beim Akzeptanztester vorbei. Er hatte eine Verständnisfrage bezüglich der Anzahl Control-Modellen in einem Projekt. Nach fünf Minuten konnte ich den Meilenstein als abgeschlossen bezeichnen.
Hilfestellungen	-
Bemerkung	Meilenstein „Entscheidugnsphase / GUI-Entwurf fertig“ abgeschlossen
Mailverkehr	Definitiver GUI-Entwurf an Auftraggeber Einladung zum Akzeptanztest an Anton Kryenbuehl: 15. April 14:00
Nächste Schritte	GUI-Oberfläche benennen und Code strukturieren (Callback-Funktionen erzeugen) Struktogramme erstellen und die Funktionen beginnen zu implementieren
Fazit	Der definitive GUI-Entwurf gefällt mir sehr gut. Ich glaube, dass ich das Erscheinungsbild, welches im ersten Moment eventuell etwas überfordert, durch geschickte Benutzerführung vergessen lassen kann. Das Ermitteln von Testfällen lernten wir nie richtig in der Schule und in den ÜKs wirkte es für mich immer sehr banal. Ich hatte keine grossen Schwierigkeiten Testfälle zu finden, jedoch denke ich man hätte besser an die Sache herangehen können, als ich es tat.

Donnerstag, 09.04.2015

Gesetzte Ziele	#401 – Oberfläche mit GUIDE erstellen inkl. Benennung #402 – Struktogramme Callback-functions #403 – Coding1: Callback-functions implementieren
Oberfläche mit GUIDE erstellen inkl. Benennung	Die Oberfläche hatte ich ja bereits vom Design her als „figure“. Das Benennen der Elemente erforderte mehr Konzentration als ich gedacht hätte und war mühsam. Die Callback-Funktionen liessen sich mit der richtigen Konfiguration einfach in das Matlab-File erzeugen, so waren sie für die Implementation vorbereitet.
Struktogramme Callback-functions	Zuerst stellte sich die Frage, für welche Funktionen ein Struktogramm entworfen werden soll. Mir fiel schnell auf, dass diverse Schritte in mehreren Funktionen vorkommen würden. So begann ich zusammenzufassen, was das Erstellen des Struktogramms für die meisten Callback-Funktionen erübrigte. Am Ende entstanden vier Struktogramme. Ich war vorerst zufrieden, war mir aber bewusst, dass möglicherweise die Struktogramme während dem Coden noch ergänzt werden müssen.
Coding1: Callback-Functions implementieren	Mir fiel schnell auf, dass die Struktogramme sehr hilfreich sind. Dennoch hatte ich kurz nach Beginn schon ein Problem, dass mir jegliches weiter Arbeiten verhinderte. Eine für mich unbekannte Fehlermeldung erschien und verunmöglichte den Funktionsaufruf der Projektauswahl-Drop-Down. Ich überlegte mir mehrmals, ob Hilfe holen besser wäre, da ich so wahrscheinlich weniger Zeit verlieren würde. Ich entschied zuerst alles Mögliche zu versuchen. Den geschrieben Code komplett auszukommentieren führte nicht zum Erfolg. Erst als ich die Callback-Funktion aller Dropdowns neu erzeugt hatte, war das Problem behoben. Ich bin nicht so weit gekommen wie geplant. In den nächsten Tagen muss ich schneller Code schreiben als heute.
Hilfestellungen	Meeting mit Auftraggeber/Fachvorgesetzem. Wir gingen auf einige Teile des IPA-Berichts ein, schauten kurz den Akzeptanztest an und besprachen grundsätzliches zu den Struktogrammen.
Bemerkung	Ich hoffe, dass ich in der restlichen Realisierungsphase von zeitfressenden Problemen verschont bleibe. Das Risiko wurde heute mit dem Auftraggeber genau besprochen. Mir sind die Konsequenzen bewusst.
Nächste Schritte	Realisieren der Callback-Funktionen, Simulation und dann White-Box-Test
Fazit	Meine Matlab-Kenntnisse könnten besser sein. Ich muss sehr oft die einfachsten Dinge in der Matlab-Hilfe nachschauen, da ich mit der Syntax nicht ausreichend vertraut bin. Ich glaube, dass ich fähig bin das GUI erfolgreich zum Laufen zu bringen, allerdings mit einer ungenügenden Qualität des Codes. Dank der Pufferzeit konnte ich schon mit der Arbeit von Morgen beginnen. Es ist allerdings schwer zu sagen, wo ich mit der Implementation aktuell stehe.

Freitag, 10.04.2015

Gesetzte Ziele	#403 – Coding1: Callback-functions implementieren #404 – Coding2: Einbinden des bestehenden Codes zur Simulierung
Coding1: Callback-Functions implementieren	Alles zu implementieren dauerte länger als ich annahm, hauptsächlich da ich sehr oft debuggen musste. Der Code beinhaltet noch Optimierungspotenzial. Ich hoffe, dass ich noch Zeit finde den Code zu verschönern. Zum Beispiel sind meine Funktions-Header bis jetzt zu wenig ausführlich. Das GUI funktioniert im Allgemeinen schon jetzt sehr gut. Was noch fehlt sind die Hilfe-Funktionen und was sich aus den White-Box-Testfällen ergibt.
Coding2: Einbinden des bestehenden Codes zur Simulierung	Vor der IPA hatte ich den Fremdcode schon sehr genau untersucht. Ich wusste also heute ziemlich genau, was alles zu tun ist. Die Herausforderung ist in erster Linie, dass der Fremdcode Rückmeldungen an mein GUI geben muss. Im Endeffekt musste ich ein bestehendes M-File an sechs stellen anpassen und zusätzlich eine neue Funktion schreiben. Ich nahm mir kurz Zeit um probemässig ein Akzeptanztest durchzuführen. Dies deckte ein paar grundsätzliche Fehler im Code auf, welche ich beheben konnte. Ausserdem konnte ich so, die Konsistenz der Testfälle überprüfen.
Mailverkehr	Aktueller Stand des IPA-Berichts an Schwester und Vater geschickt. Sie überprüfen bis nach dem Wochenende meine Sprache.
Bemerkung	Es kostete Etwas Zeit, bis ich Arbeitsjournal, Glossar usw. im Bericht integriert hatte und abschicken konnte.
Nächste Schritte	White-Box-Testfälle, GUI-optimieren, Hilfe-Funktion
Fazit	Ich fühlte mich mit meinen Matlab-Kenntnissen nicht mehr so unsicher wie gestern. Das Programmieren machte mir grossen Spass. Ich sehe den Fortschritt deutlich, wenn ich mit dem Stand von heute Morgen vergleiche. Am nächsten Tag werde ich Wege finden müssen, wie ich die Modell- und Excel- Dateien für die Simulation manipulieren kann. Ich überlege mir, deswegen die Sitzung mit dem Auftraggeber am folgenden Tag um die Mittagszeit durchzuführen. So kann ich zuerst mit meinem Wissen loslegen und anschliessend zusätzliche Tipps holen.

Montag, 13.04.2015

Gesetzte Ziele	#405 – White-Box Testfälle ermitteln #406 – Coding3: Usability Improve / Error Handling #407 – Help-Funktion erstellen
White-Box Testfälle ermitteln	Mein Leitgedanke war: Was kann der Benutzer alles schief machen? Ich denke mir, dass im Akzeptanztest die erfolgreiche Art und Weise, wie man das GUI benützt, abgedeckt ist. Deswegen werden alle Ausnahmesituationen und Fehlerszenarien im White-Box-Test geprüft. Es sind keine Texteingaben erforderlich, was das ganze vereinfacht. Ich konzentrierte mich auf das Manipulieren von Dateien oder Ordner, welche die Inputs darstellen. Ich hatte teilweise das Problem, dass ich Testfälle ermittelte, welche eigentlich den bestehenden Fremdcode testeten. Dies ist nicht Teil meiner Aufgabe.
Coding 3: Usability Improve / Error Handling	Der Hauptteil der Arbeit war das Hinzufügen von Meldungen im „else“-Teil einer „If“-Abfrage. Ich fand viele Situatinen, welche ich bemerkte, dass der Benutzer besser informiert werden müsste. Während der Verwendung des GUIs bemerkte ich immer wieder Unschönheiten, welche ich bereinigte. Im Hinterkopf behielt ich die White-Box Testfälle, welche ich vorher ermittelt hatte.
Help-Funktion erstellen	Die wichtigsten Anforderungen an ein Hilfe-Fenster sind, dass das GUI trotz geöffneter Hilfe weiter benutzt werden kann. Ausserdem muss die Hilfe auf dem Bildschirm umplatziert werden können. Ich fand eine geeignete Möglichkeit. Das Verfassen der Texte auf Englisch war zeitaufwändig. Ich glaube, dass die Texte prägnant sind.
Hilfestellung	Die Themen der Sitzung mit dem Auftraggeber waren der White-Box-Test. Ich bin keine Experte in Simulink oder TsNet. Er konnte mir die nötigen Informationen liefern.
Mailverkehr	Mail an Zweitexperte, bezüglich der Abage des IPA-Berichts
Bemerkung	Der Realisierungs-Meilenstein ist bis auf das Verfassen einiger Help-Texte somit abgeschlossen.
Nächste Schritte	Help-Funktion beenden, White-Box-Test durchführen und Anpassungen vornehmen
Fazit	Ich bin sehr zufrieden mit der IPA. Ich verspühre wenig Zeitdruck und glaube dass meine Arbeit bisher erfolgreich verlaufen ist. Die letzten drei Tage werde ich mehr oder weniger dokumentieren müssen. Ich weiss, dass dies sehr viel Zeit in Anspruch nimmt. Die neun Stunden „IPA-Bericht führen“, welche im Zeitplan verbleiben, müssen reichen.

Montag, 14.04.2015

Gesetzte Ziele	#407 – Help-Funktion erstellen #501 – White-Box-Test durchführen #703 – IPA-Bericht führen
Help-Funktion erstellen	Wenn jemand Hilfe benötigt, kann er sich sehr wahrscheinlich mit den Hilfe-Texten besser zurechtfinden. Das GUI verfügt nun über die wichtigsten Hilfeinformationen. Um ausführlichere Hilfestellungen zu verfassen, fehlte mir die Zeit. Die Hilfe-Buttons erfüllen ihren Zweck. Mir gefallen die Hilfe-Texte optisch nicht. Die Funktoin `msgbox()` erlaubt keine Formatierung der Schrift. Ich vermute, dass es noch eine bessere Möglichkeit gibt, eine Hilfe-Funktion in ein GUI zu integrieren.
White-Box-Test durchführen	Die White-Box-Tests deckten in meinem GUI viele Schwachpunkte auf. Es waren einige Code-Anpassungen notwendig. Die grösste Anpassung war im Zusammenhang mit der Tatsache, dass nicht jede Excel-Datei eine TsNet-Struktur aufweist. Ich benötigte dazu die Hilfe von meinem Auftraggeber. Er kennt TsNet sehr gut und half mir über das weitere Vorgehen zu entscheiden. Vor den Tests führte die Simulation mit einer leeren Excel-Datei in einen „Read Error“, welchen die Simulation stoppte. Mit der Erweiterung wird die Simulation nicht begonnen, wenn die Excel-Datei die Anforderungen nicht erfüllt.
IPA-Bericht führen	Ich verbesserte sprachliche Fehler mit Hilfe der Korrekturen, welche meine Schwester und meinen Vater übers Wochenende gemacht haben. Ich integrierte die Struktogramme und den White-Box-Test in den IPA-Bericht.
Hilfestellung	Der Auftraggeber half mir bei der Deutung eines fehlgeschlagenen White-Box-Testfalls. Er gab mir Ratschläge für das weitere Vorgehen. Ausserdem lieferte er mir Informationen zu TsNet.
Mailverkehr	Ich habe eine E-Mail erhalten, dass meine Nachricht an den Zweitexperten gestern nicht zugestellt werden konnte. Ich versuche Morgen den Zweitexperten per Telefon zu erreichen, falls er sich nicht meldet. Ich muss wissen, ob ich für ihn die Arbeit ausdrucken und binden muss.
Nächste Schritte	Selbständiger Akzeptanztest mit GUI Akzeptanztest mit Applikationsentwickler IPA-Bericht: Kurzfassung, Fazit, Anhänge und Bewertungskriterien lesen
Fazit	Der Code musste im Zusammenhang mit den White-Box-Tests an mehreren Stellen angepasst werden. Einerseits erstaunte mich dies und andererseits war ich froh darüber. Mein IPA-Bericht ist schon jetzt auf einem guten Stand. Ich vermute, dass die beiden letzten Tage locker verlaufen werden. Vorausgesetzt der Akzeptanztest schlägt nicht fehl.

Dienstag, 15.04.2015

Gesetzte Ziele	#501 – White-Box-Test durchführen #502 – Akzeptanz-Test durchführen #703 – IPA-Bericht führen
White-Box-Test durchführen	Alle Testfälle, welche beim ersten White-Box-Test fehlgeschlagen sind, wurden erneut durchgeführt. Dadurch konnte die Fehlerkorrektur getestet werden.
Akzeptanz-Test durchführen	Ich führte selbständig ein kompletter Durchlauf des Akzeptanztest aus. Alle Erwartungen waren erfüllt. Der Akzeptanztester war während einer dreiviertel Stunde bei mir. Er las den Testfall, führte diesen im GUI aus und füllte dann das Testergebnis aus. Am Ende bat ich ihn noch selbständig Software-Fehler zu finden. Er brachte das GUI nicht zum Absturz. Er fügte ein paar Bemerkungen im Testergebnis hinzu. Das Produkt hat den Akzeptanztest ist bestanden.
IPA-Bericht führen	Das Kapitel ‚Realisierung' habe ich bisher völlig vernachlässigt. Ich habe es als zweckmässig erachtet, ein paar Screenshots in den IPA-Bericht einzufügen. Ansonsten könnten die Experten die Help-Funktion oder das GUI während dem Gebrauch gar nicht sehen. Ich habe die Worklfow-Analyse im IPA-Bericht als Anhang hinzugefügt. Zur besseren Übersicht musste ich aber vorher die Prozesse im Flussdiagramm durchnummerieren. So kann man schneller die entsprechende Beschreibung in der Tabelle finden. Das Kapitel ‚Kontrolle' habe um den kompletten Akzeptanztest erweitert. Die Unterkapitel ‚Testablauf' und ‚Testumgebung' wurden vervollständigt.
Hilfestellung	In der Besprechung mit dem Auftraggeber habe ich folgende Themen behandelt: - Bestandteile im Kapitels ‚Realisierung' des IPA-Bericht - Ausgangslage der IPA-Kurzfassung - Beschreibung der Testumgebung Er hat mir empfohlen die fehlgeschlagenen White-Box-Testfälle in einem sogenannten Nachtest zu wiederholen.
Bemerkung	Der Zweitexperte hat mich zurückgerufen, nachdem ich ihn nicht erreicht hatte.
Nächste Schritte	IPA-Bericht: Kurzfassung beenden, Schlussfazit, Zeitplan und Journal Abschluss: drucken, binden, uploaden
Fazit	Ich hatte vor dem Testen bedenken, dass ich darüber wenig Fachwissen besass. Ich glaube, dass meine Tests zweckmässig sind und eine gute Qualität sicherstellen. Ich bin noch immer gut im Zeitplan und hoffe, dass morgen keine unvorhergesehenen Probleme die Fertigstellung meiner Arbeit behindern.

Donnerstag, 16.04.2015

Gesetzte Ziele	#703 – IPA-Bericht führen #601 – Abgabe: drucken, binden, upload
IPA-Bericht führen	Schlusswort war schnell verfasst. Die IPA-Kurzfassung bereitete mir etwas mehr Mühe, da viele Informationen in wenig Text verpackt werden mussten. Was ist von Bedeutung und was ist irrelevant? Das einfügen des Codes im Anhang war unkompliziert. Ich glaube, dass der Code verständlich ist, obwohl ich beinahe den gesamten Fremdcode weggelassen habe. Der Zeitplan und dieser Eintrag des Arbeitsjournals wurden nachträglich noch ausgedruckt.
Abgabe: drucken, binden, upload	Ich bemerkte, dass diese Tätigkeit deutlich mehr Zeit beansprucht als angenommen. Ich druckte vier Exemplare aus: Beide Experten, Fachvorgesetzer und für mich. Drei Exemplare werden gebunden und das vierte für mein Fachvorgesetzer soll in einem Ordner abgelegt werden. Der Zeitplan und dieser Eintrag des Arbeitsjournals wurden nachträglich noch ausgedruckt.
Hilfestellung	Ich bekam Hilfe beim Bedienen der Binde-Maschine. Ich musste nachschauen, bis wann die IPA hochgeladen werden muss und wie das die Deckblätter eingesetzt werden müssen. Den IPA-Bericht habe ich kurz von einem Arbeitskollegen auf die sprachliche Korrektheit überprüfen lassen.
Nächste Schritte	Nach IPA: WebSummary, Vorbereitung auf IPA-Demo
Fazit	Ich machte mich gefasst auf eine Menge ‚Sisyphusarbeit' am letzten Tag. Ich bin pünktlich fertiggeworden, obwohl ich den Aufwand der beiden geplanten Tätigkeiten unterschätz habe.

6. IPA-Kurzfassung

6.1 Ausgangssituation

Das Tool IMSES mit Basis Matlab / Simulink (R) ermöglicht unteranderem das Simulieren von Applikationssoftware für die Gebäudeautomation. Die Umsetzung des sogenannten Open-Loop-Tests in IMSES hat grosses Optimierungspotenzial. Der Workflow von diesem Use Case wurde im Vorfeld der IPA analysiert. Die Ergebnisse liegen in Form eines Flussdiagramms mit Beschreibung vor. Ziel der IPA ist ein einheitliches GUI mit zweckmässigem Design, Implementierung der Funktionen und abschliessendem Testing.

6.2 Umsetzung

Das Vorgehen geschah nach der Projektmanagement-Methode „IPERKA“. In der Planungsphase wurde ein Zeitplan über die zehn Tage erstellt und dabei fünf Meilensteine definiert: Planung, Entscheidung, Realisierung, Kontrolle und Abschluss

In der Entscheidungsphase wurde zuerst ein GUI-Konzept ermittelt. Das GUI sollte dem Workflow entsprechend übersichtlich gliedert sein. Die beste Möglichkeit zur Gliederung wurde mit einer Nutzwertanalyse bestimmt. Das GUI soll den gesamten Inhalt in einem Fenster vereinen. Pro Workflow-Schritt fasst ein Panel die Buttons, Textfelder usw. zusammen. Das Prinzip der Hilfe-Funktion und der Benutzerinformation wurde ebenfalls im GUI-Konzept festgelegt: Hinter jedem Panel steht ein Button mit der Aufschrift „?“ und eine Info-Box.

Im anschliessenden GUI-Entwurf wurden die Workflow-Schritte im GUI-Konzept integriert. Im ersten Schritt „Test-Project-Selection“ kann man ein Projekt auswählen. Das Projekt sammelt die Daten für den Open-Loop-Test in einer Ordnerstruktur. In den nächsten beiden Schritten werden ein Control-Model, welches die Applikationssoftware aufweist, und eine TsNet-File angewählt. Es muss zusätzlich definiert werden, welches TsNet-Sheet den Testcase beinhaltet.

Vor der Realisierung wurde der GUI-Entwurf von einem Mitarbeiter validiert. Ausserdem wurden Akzeptanz-Testfälle definiert, welche er in der Kontrollphase ausführt. Während der Realisierung wurden White-Box-Testfälle ermittelt. Der White-Box-Test wurde direkt nach der Realisierung vom IPA-Ausführenden durchgeführt.

6.3 Ergebnis

Das GUI verfügt über 20 Buttons, zwei Drop-Down-Menus, vier Info-Boxen und fünf Textfelder. Um stets gute Orientierung auf dem GUI zu gewährleisten, werden GUI-Elemente, die nicht verwendet werden dürfen, deaktiviert. Die Textfelder sind ausschliesslich durch Buttons veränderbar. Nach jeder Benutzerinteraktion wird eine Funktion durchlaufen, welche den Fortschritt des Workflows überprüft und die GUI-Elemente dementsprechend (de)aktiviert.

Der Hauptteil des Codes (Callback-Funktionen mit Unterfunktionen) befindet sich in der Startdatei „IMSES_GUI.m“. Für die Simulation des Open-Loop-Tets wird ein bestehender Fremdcode aufgerufen, der an wenigen Stellen angepasst werden musste. Die Anpassungen stehen im Zusammenhang mit der Kommunikation von Statusmeldungen.

Das entstandene Tool dürfte von den Entwicklern ziemlich selbsterklärend benutzt werden können. Die Help-Funktion hilft beim Verständnis des GUIs und Benutzereingaben. Auf eine Benutzeranleitung wird verzichtet.

7. Entscheidung

7.1 GUI-Konzept

7.1.1 Erarbeiten der Möglichkeiten

7.1.1.1 Einleitung

Die Elemente des GUIs wie zum Beispiel Buttons und Textfelder sind in einem Fenster platziert. Wenn sich zu viele Elemente auf engstem Raum befinden, verliert man schnell die Übersicht. Darum ist es sinnvoll die GUI-Elemente zu gruppieren. Die entstandenen Gruppen bilden die Workflow-Schritte. Womit die Workflow-Schritte am zweckmässigsten gegliedert werden können, wird in diesem Kapitel erarbeitet.

7.1.1.2 Möglichkeit 1: Mehrere Fenster (Buttons zur Navigation)

Es wird ermöglicht via Buttons zwischen den einzelnen Schritten hin und her zu wechseln. Hier wird die Möglichkeit angedeutet:

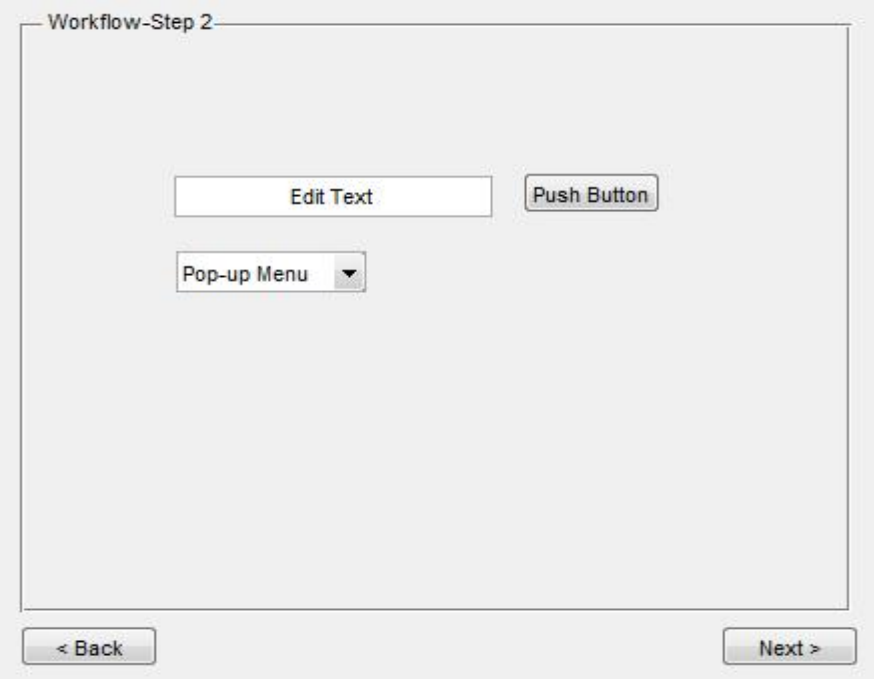


Abbildung 7-1: Beispiel Möglichkeit 1

Durch den Klick auf einen der zwei Buttons unten, wird das andere Fenster gezeigt und das Aktuelle ist nicht mehr sichtbar.

7.1.1.3 Möglichkeit 2:Tabs

Tabs ermöglichen es, alle Inhalte in ein Fenster zu integrieren. Das Symbolbild zeigt wie diese Möglichkeit aussehen könnte:

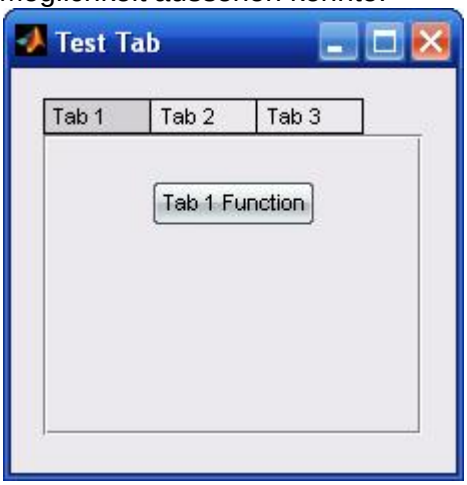


Abbildung 7-2: Beispiel Möglichkeit 2 [3]

Der Nachteil bei dieser Lösung ist, dass „Matlab GUIDE“ in der verwendeten Version diese Funktion nicht enthält. Die Tabs folglich müssten im Code festgelegt werden.

7.1.1.4 Möglichkeit 3: Alles in einem Fenster (mit Panels)

Mit Panels kann man GUI-Elemente optisch trennen oder zusammenfassen. Das Bild veranschaulicht die Möglichkeit, wenn alle Panels in ein Fenster gepackt werden:

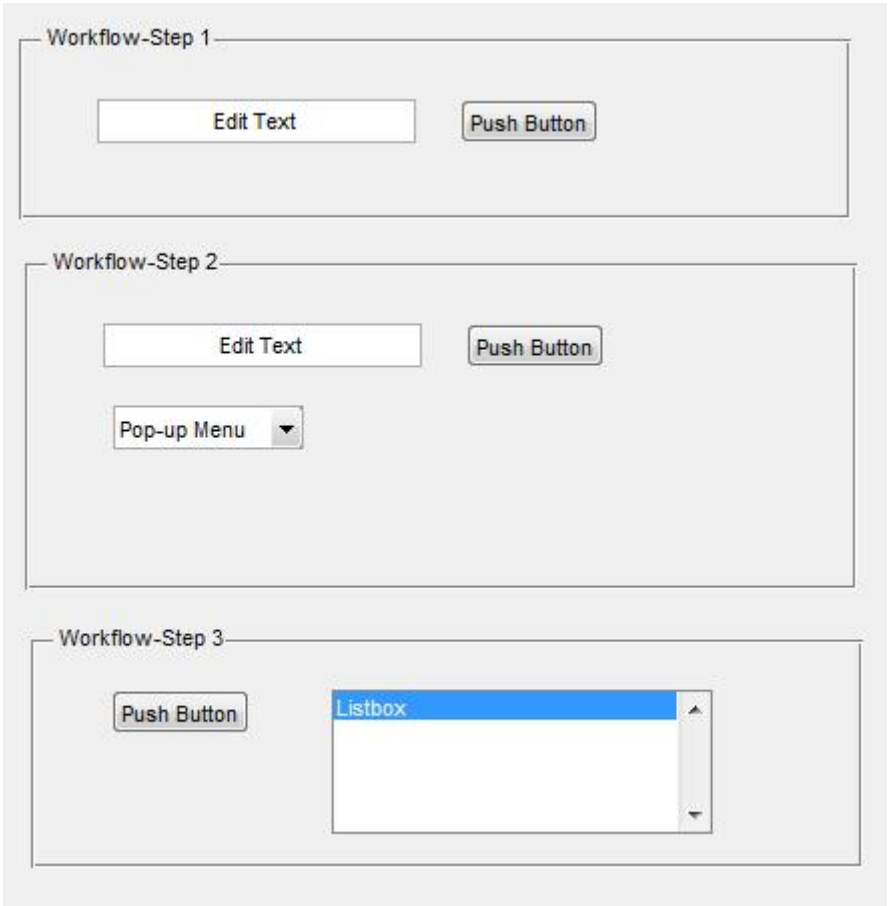


Abbildung 7-3: Beispiel Möglichkeit 3

7.1.1.5 Kriterien für das GUI-Konzept

Kriterium	**Beschreibung**
Realisierungsaufwand	Welchen Aufwand muss für den Entwickler zur Umsetzung betrieben werden?
Wenig User-Interaktion	Kann ohne User-Interaktion möglichst viel Inhalt erfasst werden? Müssen zwischen den Schritten unnötige Klicks gemacht werden?
Orientierung/Übersicht	Besteht die Gefahr, dass der Benutzer die Übersicht verliert oder überfordert ist?

Tabelle 7-1: Kriterien

Um das optimale GUI-Konzept zu finden, werden jetzt alle Gliederungs-Möglichkeiten auf diese Kriterien überprüft:

Möglichkeit 1: Mehrere Fenster (Buttons zur Navigation)	
Realisierungsaufwand	Das Erstellen der Fenster ist ein eher kleiner Aufwand. Jedoch muss der reibungslose Datenaustausch zwischen den Fenstern gewährleistet sein. = *Mittelmässiger Aufwand*
Wenig User-Interaktion	Es sind viele Klicks erforderlich, was nicht erwünscht ist. Der Nachteil wird vor allem ersichtlich, wenn man zwischen mehreren Schritten navigieren muss. = *zu viel User-Interaktion*
Orientierung/Übersicht	Da nur ein Minimum an Inhalt gleichzeitig sichtbar ist, besteht die Gefahr nicht, dass der Benutzer die Übersicht verliert. Die Orientierung im Workflow beizubehalten ist dafür etwas schwieriger. = *optimale Übersicht, eher schlechte Orientierung*

Tabelle 7-2: Möglichkeit 1

Möglichkeit 2: Tabs	
Realisierungsaufwand	Wie bereits erwähnt, können Tabs nicht graphisch dem GUI hinzugefügt werden. Für mehr Code muss mehr Zeit investiert werden. = *grosser Aufwand*
Wenig User-Interaktion	Es muss lediglich zwischen den Tabs hin und her navigiert werden. Jeder Tab ist zu jedem Zeitpunkt verfügbar. = *nicht zu viel User-Interaktion*
Orientierung/Übersicht	Die einzige Gefahr ist, dass zwischen den Tabs die Übersicht verloren geht. Innerhalb eines Tabs ist nur wenig Inhalt enthalten, was die Orientierung unkompliziert macht. = *gute Übersicht, mittlere Orientierung*

Tabelle 7-3: Möglichkeit 2

Möglichkeit 3: Alles in einem Fenster (Panels)	
Realisierungsaufwand	Geringer Realisierungsaufwand, da nur in einem Fenster gearbeitet wird. =*kleiner Aufwand*
Wenig User-Interaktion	Es sind keine zusätzlichen User-Interaktionen nötig, da zu jedem Zeitpunkt alle Inhalte ersichtlich sind. = *keine unnötige User-Interaktion*
Orientierung/Übersicht	Die Wahrscheinlichkeit, dass Orientierungsschwierigkeiten auftreten ist relativ hoch. Viel Inhalt bedeutet schlechte Übersicht. = *eher schlechte Übersicht*

Tabelle 7-4: Möglichkeit 3

7.1.1.6 Nutzwertanalyse

Die Nutzwertanalyse soll helfen die beste Möglichkeit zu finden. Dazu werden den Kriterien Gewichtungen zugewiesen. Mit 45% ist der Realisierungsaufwand das wichtigste Kriterium, da nicht viel Zeit für die Implementierung zur Verfügung steht. Gute Übersicht ist im GUI wichtiger als wenig User-Interaktion. Die Bewertung erfolgt in Ganzzahlen:

- 1 = schlecht
- 6 = optimal

	Gewicht	**Möglichkeit 1: mit Buttons**		**Möglichkeit 2: Tabs**		**Möglichkeit 3: Panels**	
Kriterium	(0-100%)	Bewertung	Ergebnis	Bewertung	Ergebnis	Bewertung	Ergebnis
Realisierungs-aufwand	45%	3	1.35	1	0.45	5	2.25
Wenig User-Interaktion	20%	1	0.2	4	0.8	6	1.2
Orientierung/ Übersicht	35%	6	2.1	5	1.75	2	0.7
	100%		**3.65**		**3**		**4.15**

Tabelle 7-5: Nutzwertanalyse

7.1.2 GUI-Konzept-Lösung

Die Nutzwertanalyse hat ergeben, dass die Lösung mit Panels in einem Fenster am besten geeignet ist. Um die schlechte Übersicht, zu entschärfen, werden folgende Bestimmungen gestellt:

- Der Standpunkt im Workflow muss jeder Zeit ersichtlich sein (signalisiert werden).
- Deaktivierung der GUI-Elemente (oder ganzer Panels), welche von einem vorherigen Schritt abhängig sind, falls dieser Schritt noch nicht abgeschlossen ist.
- Die Hilfe-Funktion enthält sowohl Informationen zum Gesamt-Workflow, wie auch zum einzelnen Schritt.
- Das Fenster darf nicht grösser als der Bildschirm eines Notebooks sein:
  - 15 Zoll Diagonale
  - 1024 x 768
- Die Buttons und Textfelder sollen die in Windows üblichen Grössen haben.

Das GUI-Konzept wurde aufgrund der Bestimmungen folgendermassen erweitert:

- Jedes Panel hat ein zugehöriges Kästchen, welches den Abschluss des Workflow-Schrittes mit grüner Farbe und „OK“ signalisiert.
- Der Button „Help“ wurde oben rechts platziert, da beinahe alle Microsoft Produkte diesem Konform entsprechen. Der Button öffnet ein Fenster mit Informationen zum Gesamt-Workflow.
- Jeder Workflow-Schritt besitzt ein Button „?“ als spezifische Hilfestellung. Die Aufteilung in mehrere Hilfe-Buttons hat den Vorteil, dass der jeweilige Hilfe-Text kurz gehalten ist und implizit klar ist, zu welchem Schritt der Hilfe gehört.
- In den späteren Schritten sind GUI-Elemente gegebenenfalls deaktiviert.

Das folgende Bild zeigt das GUI-Konzept, ohne die späteren Inhalte, mit den vorher definierten Erweiterungen.

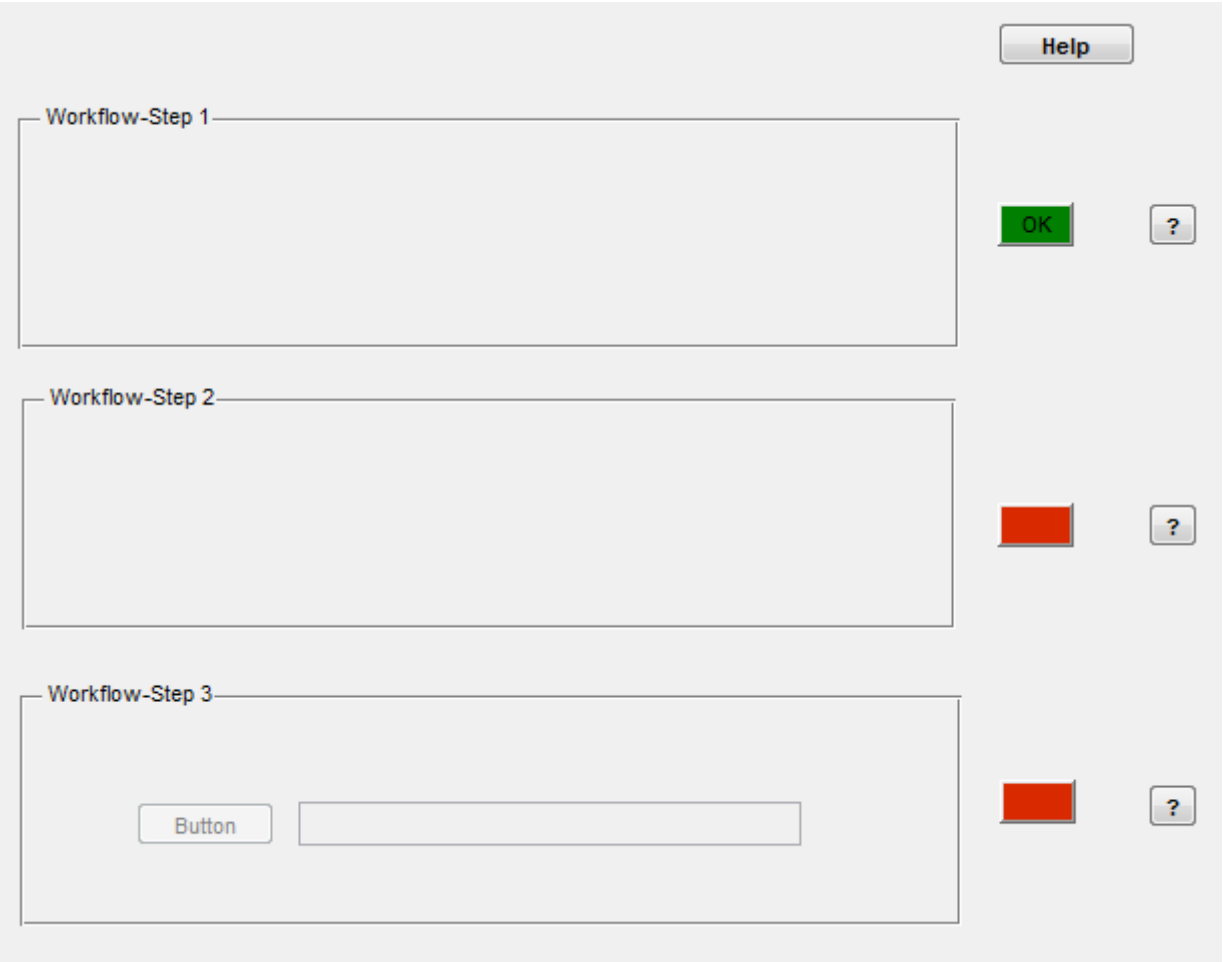


Abbildung 7-4: GUI-Konzept

7.2 GUI-Entwurf erstellen

Jetzt geht es darum die einzelnen Workflow-Schritte in die Panels zu bringen. In einem ersten Versuch sieht dies folgendermassen aus:

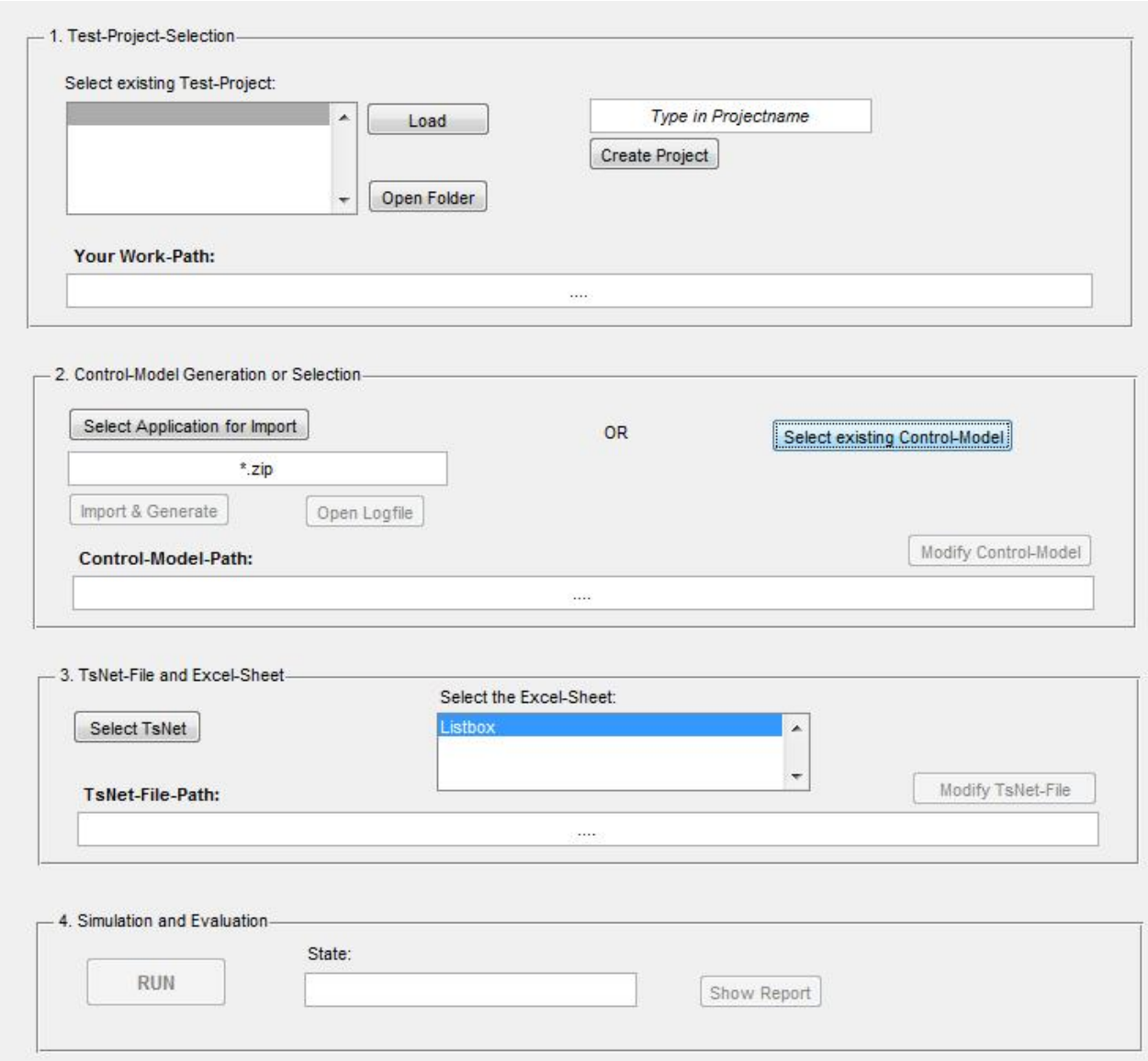


Abbildung 7-5: GUI-Entwurf 1

Eine Beschreibung der Workflowschritte (auch Panels):

1. Ein Projekt sammelt die Daten, welche für einen Test benötigt werden. Die Daten sind Control-Model und TsNet-File, welche in Schritt zwei und drei ausgewählt werden. Ein bestehendes Projekt kann ausgewählt und geladen werden. Ein neues Projekt entsteht, wenn ein Name im Textfeld eingegeben ist. Das neue Projekt wird automatisch in der Listbox hinzugefügt und angewählt. Der Pfad des aktiven Projekts wird unten angezeigt und kann durch „Open Folder“ im Windows-Explorer geöffnet werden.
2. Hier gibt es zwei Möglichkeiten:
    - Das Control-Model wird erzeugt in dem man einen ABT-Export (ZIP-File) importiert. Aus dem Import wird automatisch das Control-Model generiert.
    - Man hat bereits von früher ein Control-Model und kann es auswählen.

Im Feld „Control-Model-Path“ wird die generierte oder ausgewählte Datei angezeigt. Durch „Modify Control-Model“ kann man sie in Simulink bearbeiten.

3. Das TsNet-File stellt die Testspezifikation dar. Es enthält im entsprechenden Sheet eine Test-Script.
4. Mit Run wird die Simulation durchgeführt und während dem Statusmeldungen mit erläutert. „Show Report“ zeigt am Ende die Auswertung des Tests.

7.3 GUI-Entwurf optimieren

Die erste Version, in der Grafik auf der vorherigen Seite, muss auf die Korrektheit und verschiedene Details überprüft werden.

7.3.1 Test-Project-Selection

- Es wurde entschieden die Listbox durch ein Drop-Down-Menu zu ersetzen. Diese Änderung befürwortet:
  - Im Drop-Down-Menu sind mehr Projekte gleichzeitig sichtbar.
  - Ein Drop-Down-Menu nimmt weniger Platz weg.
  - Wenn ein Projekt ausgewählt worden ist, sind die anderen Projekte nicht mehr von Interesse. Sie müssen deshalb nicht sichtbar sein.
  - Das Label „Select existing Test-Project“ ist nicht mehr notwendig, da nun die Startauswahl des Drop-Down-Menus diese Bezeichnung trägt.
- Das Textfeld für den Projektnamen kann entfernt werden, wenn man beim Klick auf den Button „Create Project“ ein Eingabefenster erscheinen lässt. Je weniger GUI-Elemente nötig sind, desto besser ist die Übersicht.
- Es wurden die Buttons „Delete Project“ und „Save Project As“ hinzugefügt:
  (beide Buttons / Use Cases sind nicht im Workflow-Diagramm enthalten)
  - „Delete Project“ löscht ein geladenes Projekt. Es muss zusätzlich sichergestellt werden, ob sich der Benutzer sicher ist. Wenn kein Projekt geladen wurde, ist der Button deaktiviert.
  - „Save Project As“ ermöglicht ein geladenes Projekt unter einem anderen Namen abzuspeichern. Beim Klick auf den Button, wird ein Projektname in einem Eingabefenster abgefragt.

7.3.2 Control-Model Generation or Selection

- Der Benutzer kann entweder eine Applikation importieren und ein Control-Model generieren, oder ein bestehendes Control-Model auswählen. Um dies noch eindeutiger darzustellen, wurde um die erste Möglichkeit ein Panel gelegt.

7.3.3 TsNet-File and Excel-Sheet

- Der Button “Select TsNet” hat neu die Bezeichnung “Select TsNet-File”.
- Die Listbox wurde aus denselben Gründen wie in Schritt eins „Test-Project-Selection“ durch ein Drop-Down-Menu ersetzt

7.3.4 Button Aktivieren/Deaktivieren

Durch das Deaktivieren von Buttons, können viele Fehler vermieden werden. Wann welcher Button aktiviert sein soll, wird im folgenden Flussdiagramm dargestellt.

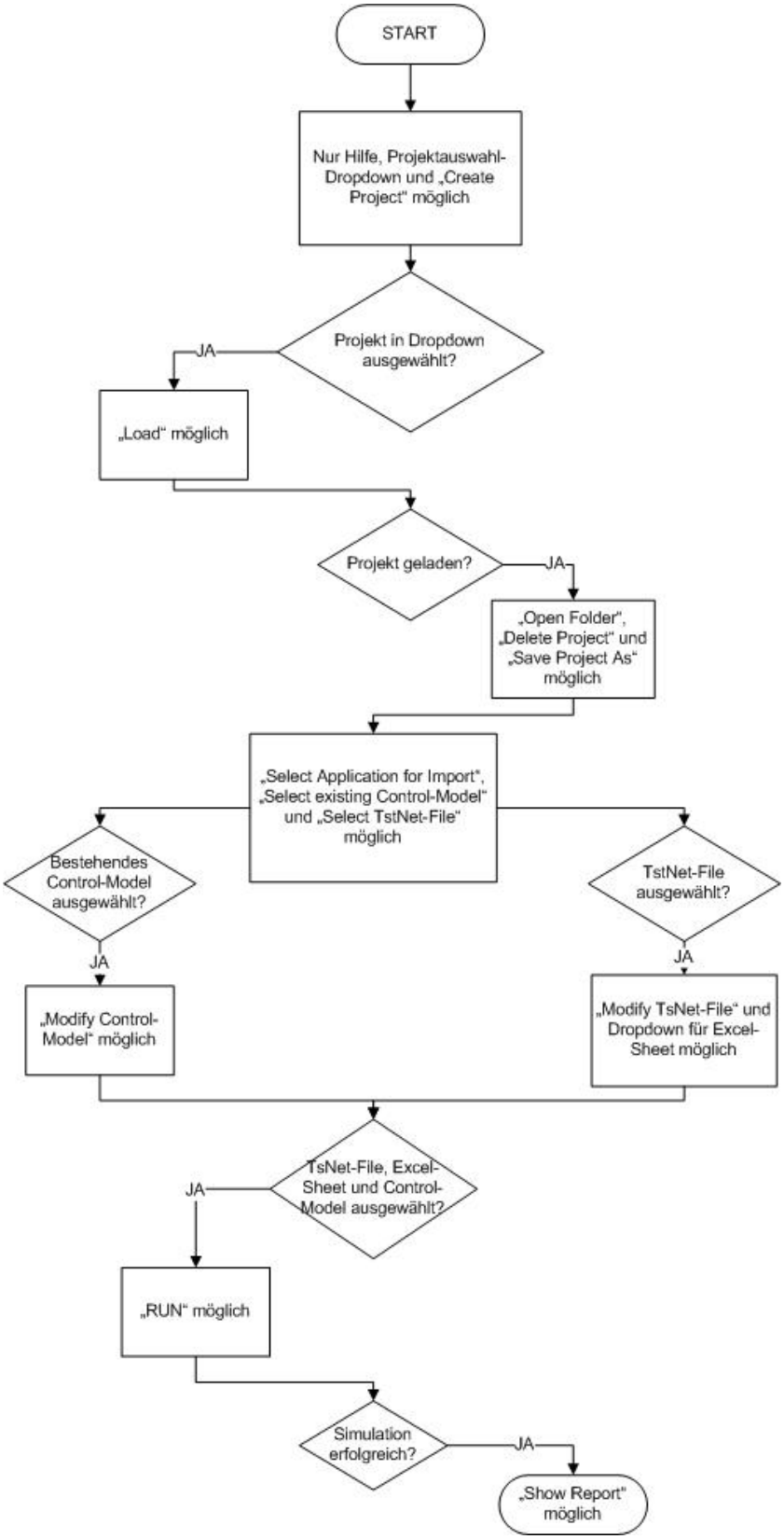


Abbildung 7-6: Button-Abhängigkeiten

7.3.5 GUI-Entwurf fertiggestellt

Unter Berücksichtigung der Optimierung steht nun der definitive GUI-Entwurf fest. Hier das Erscheinungsbild beim Öffnen:

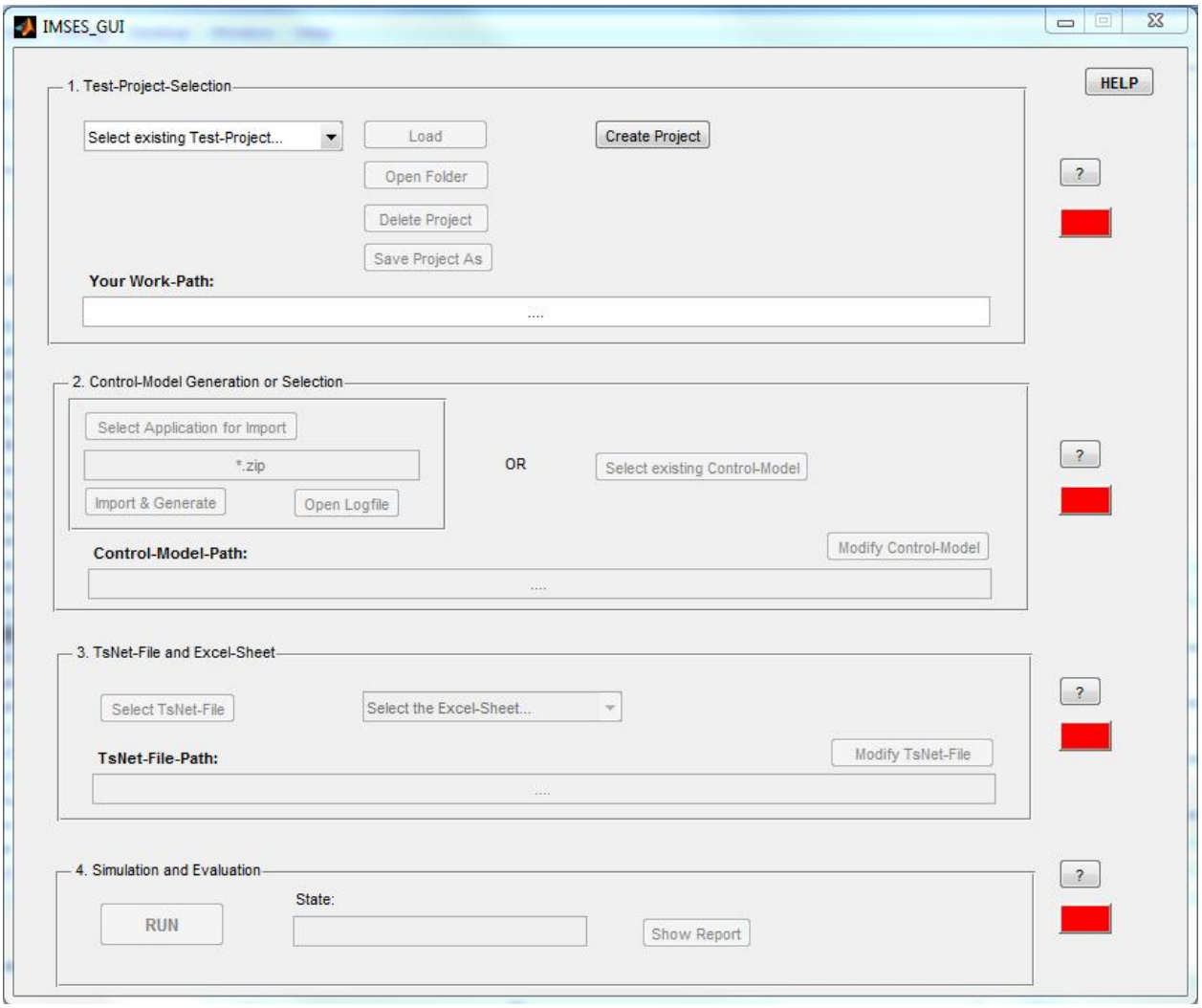


Abbildung 7-7: GUI-Entwurf defintiv

8. Realisierung

8.1 Struktogramme

Im Umfang der IPA ermöglicht das GUI, abgesehen von der Hilfe Funktion, neun verschiedene Interaktionsmöglichkeiten. Die Konsequenzen folgender GUI-Elemente sind trivial:

- Show-Report: öffnet Excel-Sheet
- RUN: startet Fremdcode
- Modify TsNet-File: öffnet Excel-File
- Modify Control-Model: öffnet Simulink-Model
- Select existing Control-Model: öffnet Fenster zur Auswahl eines Modells
- Drop-Down-Menu: Select the Excel-Sheet…
- Drop-Down-Menu: Select existing Test-Project…

Für die komplexeren Interaktionen wird ein Struktogramm erstellt:

- Load
- Select TsNet-File

Zusätzliche Funktionen, welche in einem Struktogramm dargestellt werden, sind:

- OpeningFctGUI: Wird beim Öffnen des GUIs abgearbeitet.
- updateGUI: Überprüft den Fortschritt, (de)aktiviert Buttons und kommuniziert mit dem Benutzer. Die Funktion wird am Ende jeder Interaktion aufgerufen.

8.1.1 LoadProj

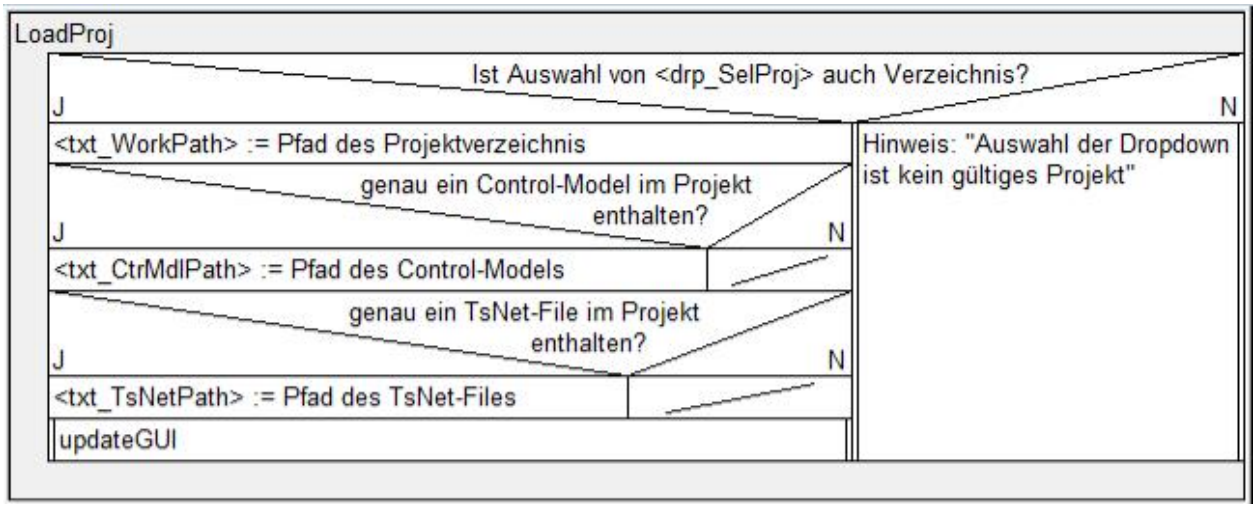


Abbildung 8-1: LoadProj

8.1.2 SelTsNet

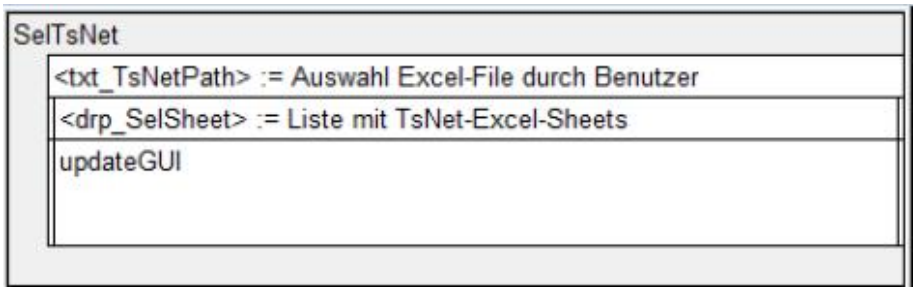


Abbildung 8-2: SelTsNet

8.1.3 OpeningFct_GUI

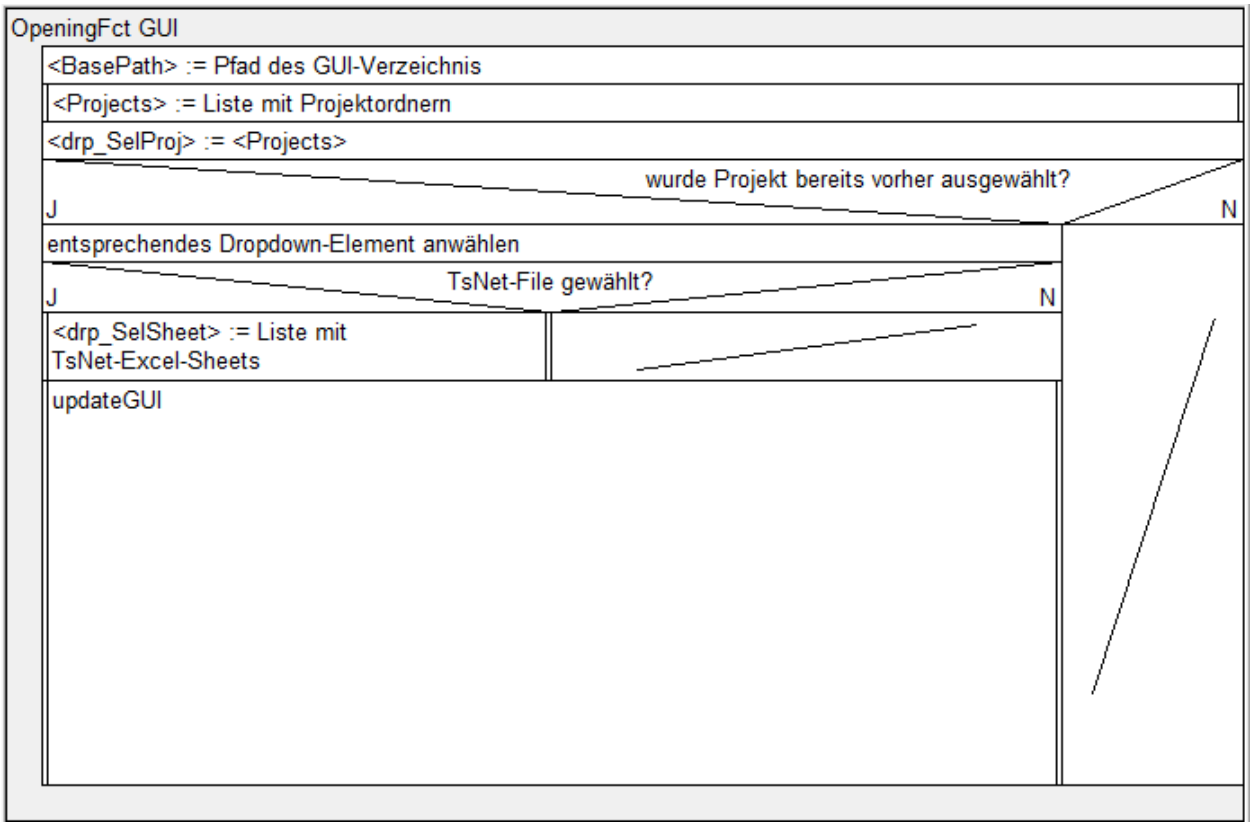


Abbildung 8-3: OpeningFct_GUI

8.1.4 updateGUI

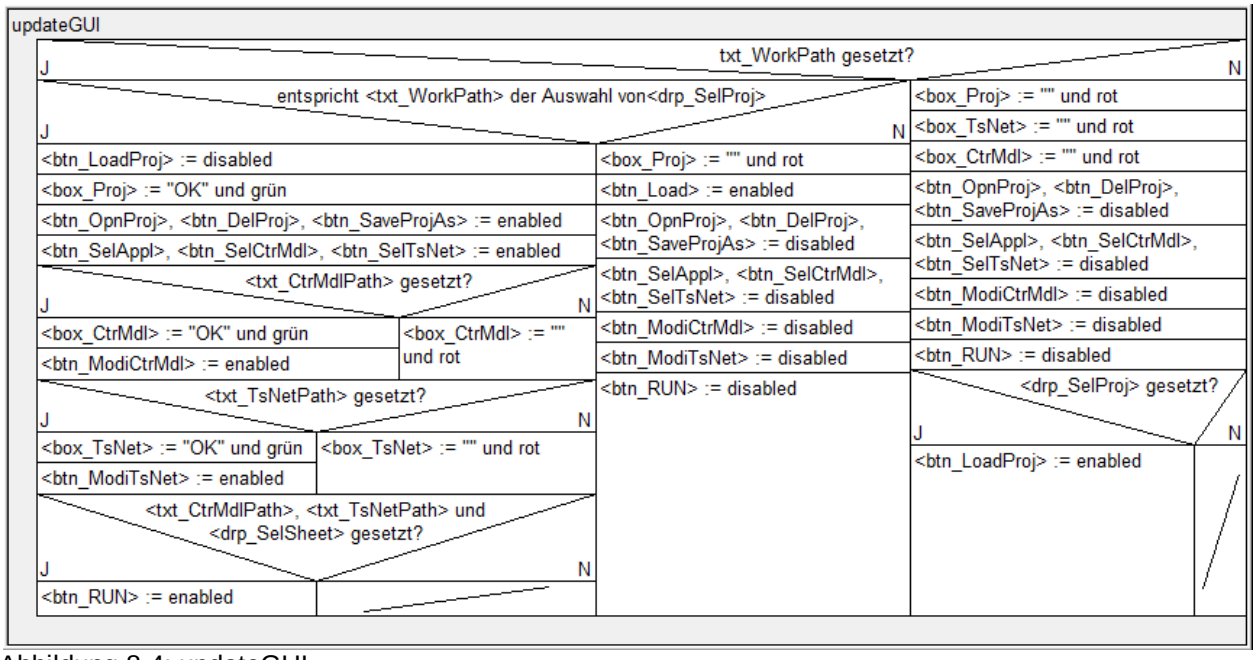


Abbildung 8-4: updateGUI

Erklärung:
<> umschliessen GUI-Elemente
(btn = Button, box = Info-Box, drp = Drop-Down-Menu)
:= definiert gleich

8.2 Umsetzung in MATLAB

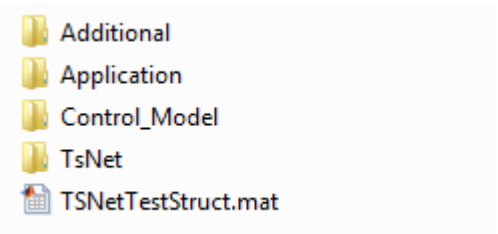
8.2.1 Dateistruktur

- Der Ordner, der diese Dateien beinhaltet, kann an einem beliebigen Ort innerhalb der MATLAB-Installation platziert werden.
- Um das GUI zu öffnen muss die Datei „IMSES_GUI.m“ ausgeführt werden (Run in Matlab).
- “IMSES_GUI.fig” ist das GUI, welche mit GUIDE erstellt wurde
- “IMSES_GUI.m” beinhaltet den Code, unterteilt in:
    - Callback-Functions: werden von GUI-Elementen aufgerufen
    - Additional-Functions: Wiederkehrende Abläufe, werden in Callback-Functions aufgerufen
- In „GuiConstants.m“ wurden alle Konstanten festgelegt.
- Im Ordner „functions“ befindet sich der bestehenden Fremdcode. Dieser Code ist fähig ein Gebäudeautomation zu simulieren und ein Report zu erstellen. Zusätzlich wurde „checkErr.m“ entwickelt. (als Schnittstelle für Fehlermeldungen zum neuen IMSES-GUI)

Sämtlicher Code kann im Anhang (Kapitel 11.2) nach gelesen werden.

8.2.2 Projektverwaltung

Die Test-Projekte werden im Ordner „Projects“ angelegt. Jeder Ordner innerhalb dieses Ordners wird als Test-Projekt betrachtet, wenn er eine festgelegte Struktur aufweist:



Ein Name des Test-Projekts stimmt mit dem Ordnernamen überein.

Die ZIP-Datei einer Applikation wird nach „Application“ kopiert. Wenn ein Import durchgeführt wird, befindet sich danach das Applikcation-Model ebenfalls dort.
„Control_Model“ beinhaltet das generierte Control-Model, welches für die Simulation verwendet wird. Achtung: Wenn manuell Änderungen vorgenommen werden, kann allenfalls die Referenz auf das Applikations-Model geändert werden.

Das Excel-File befindet sich im Ordner „TsNet“. Der Report ist ein Excel-Sheet, welches automatisch dieser Datei hinzugefügt wird.

„Additional“ kann für benutzerdefinierte Zwecke verwendet werden.

8.2.3 Screenshot des GUIs in Verwendung

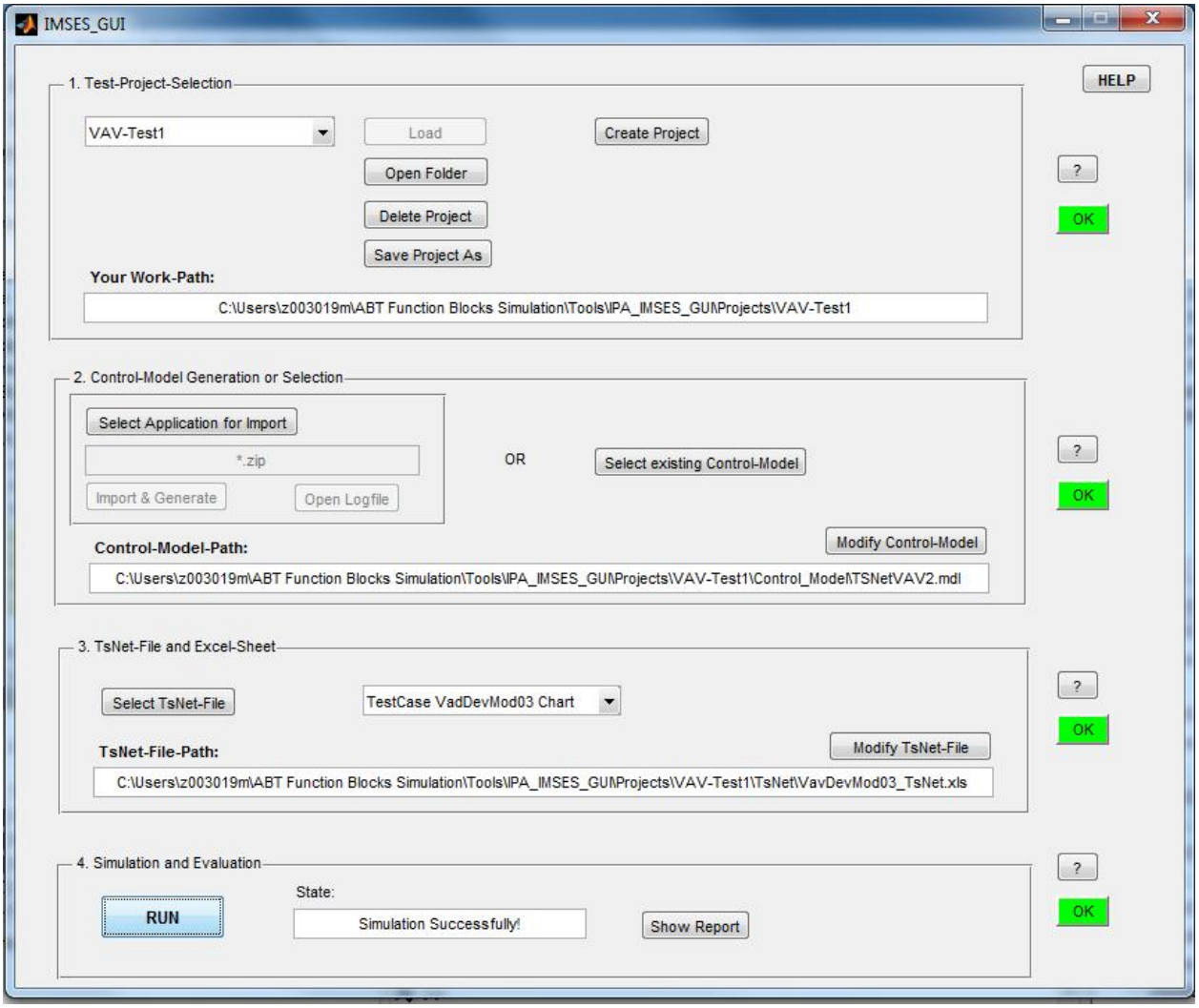


Abbildung 8-5: GUI in Verwendung

8.2.4 Screenshot der Help-Funktion

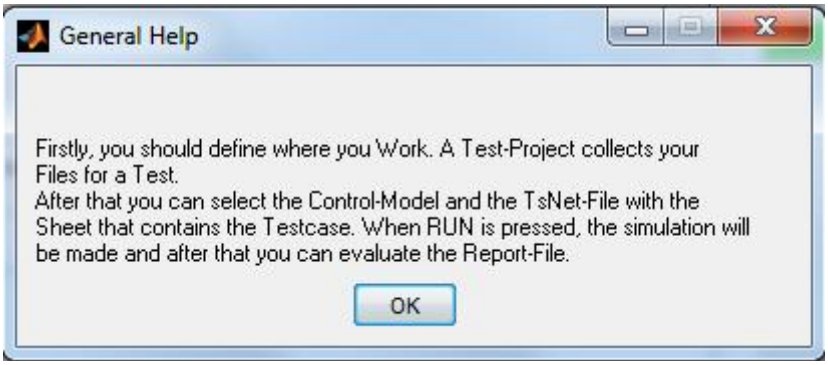


Abbildung 8-6: General Help

Abbildung 8-7: Help Test-Project

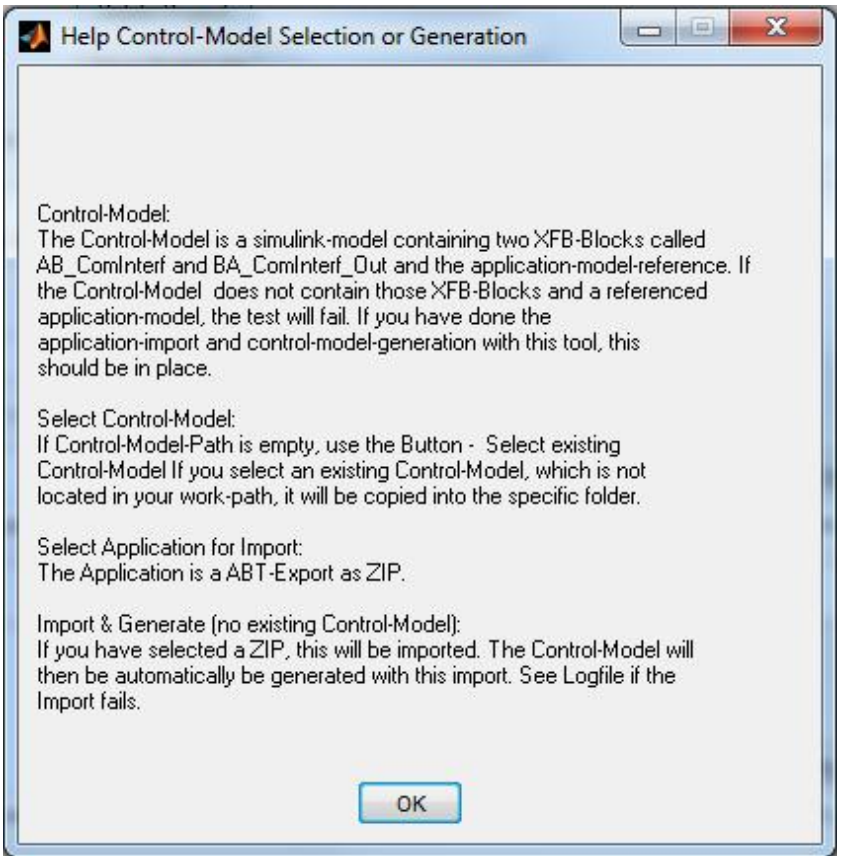


Abbildung 8-8: Help Control-Model

Abbildung 8-9: Help TsNet

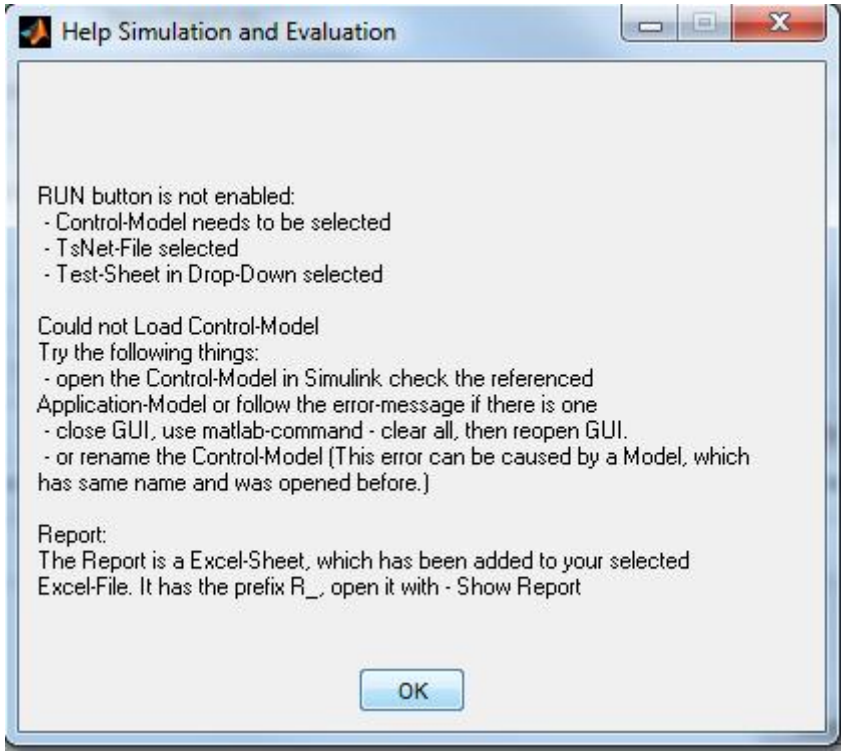


Abbildung 8-10: Help Run

9. Kontrolle

9.1 Testumgebung

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

Element	Wert
Betriebssystemname	Microsoft Windows 7 Enterprise
Version	6.1.7601 Service Pack 1 Build 7601
Zusätzliche Betriebssystembesc...	Nicht verfügbar
Betriebssystemhersteller	Microsoft Corporation
Systemname	MD12TJKC
Systemhersteller	FUJITSU
Systemmodell	CELSIUS H700
Systemtyp	x64-basierter PC
Prozessor	Intel(R) Core(TM) i7 CPU M 620 @ 2.67GHz, 2667 MHz, 2 Kern(e), 4 logis...
BIOS-Version/-Datum	FUJITSU // Phoenix Technologies Ltd. Version 1.18, 26.10.2010
SMBIOS-Version	2.6
Windows-Verzeichnis	C:\WINDOWS
Systemverzeichnis	C:\WINDOWS\system32
Startgerät	\Device\HarddiskVolume2
Gebietsschema	Vereinigte Staaten von Amerika
Hardwareabstraktionsebene	Version = "6.1.7601.17514"
Benutzername	WW002\z003019m
Zeitzone	Mitteleuropäische Sommerzeit
Installierter physikalischer Speic...	8.00 GB
Gesamter realer Speicher	7.86 GB
Verfügbarer realer Speicher	4.51 GB
Gesamter virtueller Speicher	15.7 GB
Verfügbarer virtueller Speicher	11.7 GB
Größe der Auslagerungsdatei	7.86 GB
Auslagerungsdatei	C:\pagefile.sys

Abbildung 9-1: Testumgebung

9.1.1 MATLAB-Installation

Es wird mit einer Standardinstallation von MATLAB gearbeitet, was eine gültige Lizenz erfordert. Die MATLAB-Version ist: R2011b (7.13.0.564), 32-Bit (win32) vom 14. August 2011

9.1.2 IMSES-Installation

IMSES wurde am 6. Januar 2015 vom Server heruntergeladen und konfiguriert.

9.1.3 Testdaten

„TSNetVAV.mdl“ mit „Rm1.mdl (stammt aus „Segment_208_1.zip“).
„VavDevMod03_TsNet.xls“ mit dem Sheet: „TestCase VadDevMod03 Chart“

Die Testdaten wurden vor der IPA erfolgreich auf ihr Zusammenspiel geprüft.

9.2 Testablauf

Der Test setzt sich aus zwei Komponenten zusammen:

- Die White-Box-Testfälle wurden während der Realisierungsphase vom IPA-Ausführenden ermittelt. In der Kontrollphase werden die Tests von derselben Person durchgeführt.
- Die Akzeptanz-Testfälle wurden am Ende der Entscheidungsphase vom IPA-Ausführenden ermittelt. In der Kontrollphase werden die Tests von einem Mitarbeiter durchgeführt. Er erhielt eine knappe Einführung durch den Auftraggeber. Ihm wurde zusätzlich ein Test-Protokoll zur Verfügung gestellt, welches er ausfüllen soll.

Folgende Workflow-Schritte sind nicht Teil der IPA und sind daher nicht mit Testfällen abgedeckt:

- Die Funktion der Buttons: „Create Project“, „Delete Project“, „Save Project As“
- Der Import/Generation-Zweig mit dem Button „Select Application for Import“

9.3 White Box-Test

9.3.1 Testfälle

Test ID: 1	
Name	Projekt laden mit Standardauswahl
Testvoraussetzungen	Mindestens ein Projekte vorhanden
Testablauf	1. Projekt in Drop-Down-Menu auswählen 2. Klick auf „Load“ 3. „Select existing Test-Project…“ auswählen
Erwartetes Resultat:	Die Auswahl des Drop-Down-Menus sollte nicht geladen werden können. Der Load-Button sollte nicht aktiviert sein.

Test ID: 2	
Name	Unechtes Projekt
Testvoraussetzungen	-
Testablauf	1. In „Projects“ ein Ordner mit beliebigem Namen anlegen. 2. GUI öffnen 3. Auswahl des vorher vergebenen Namens in Drop-Down 4. Klick auf „Load“
Erwartetes Resultat:	Die Auswahl darf nicht geladen werden können. Das GUI bleibt unverändert. Der „Load“-Button ist aktiviert und lässt, wenn er angeklickt wird, eine entsprechende Meldung erscheinen.

Test ID: 3	
Name	Control-Model verändern
Testvoraussetzungen	Ein Projekt mit Control-Model
Testablauf	1. Projekt laden (Control-Model wird angezeigt) 2. Die Datei im Ordner „Control_Model“ umbenennen oder löschen.

	3. Klick auf „Modify Control-Model“
Erwartetes Resultat:	Die Datei sollte nur bearbeitet werden können, wenn die Anzeige mit dem Befund im Verzeichnis übereinstimmt. Ansonsten sollte das Öffnen verweigert werden mit dem entsprechenden Hinweis. Nach erneutem Laden des Projekts sollte die Datei, falls sie gelöscht wurde nicht mehr angezeigt werden und sonst mit dem neuen Namen.

Test ID: 4	
Name	Dateiauswahl umgehen mit „All Files“
Testvoraussetzungen	Ein Projekt mit Control-Model
Testablauf	1. Projekt laden 2. Klick auf „Select existing Control-Model“ 3. Dateityp auf „All files“ umstellen 4. Beliebige Nicht-Modell-Datei auswählen 5. Klick auf „Öffnen“
Erwartetes Resultat:	Die Datei mit dem falschen Dateityp darf weder im GUI angezeigt, noch ins Projektverzeichnis kopiert werden. Der Benutzer sollte über die Umstände informiert werden.

Test ID: 5	
Name	Korrupte MDL-Datei
Testvoraussetzungen	Ein Projekt
Testablauf	1. Leere Datei mit der Endung „.mdl“ im Ordner „Control_Model“ erstellen 2. Projekt und Model laden und dann auf „RUN“ klicken
Erwartetes Resultat:	Entsprechende Meldung im Status-Textfeld

Test ID: 6	
Name	Datei-Auswahl abbrechen
Testvoraussetzungen	Projekt mit Control-Model
Testablauf	1. Projekt auswählen und laden 2. Klick auf „Select existing Control-Model“ und dann auf „abbruch“
Erwartetes Resultat:	Bisherige Auswahl steht immer noch im Textfeld (keine Änderung im GUI)

Test ID: 7	
Name	Test 3,4,5 und 6 mit TsNet-File anstatt Control-Model
Testvoraussetzungen	Ein Projekt mit TsNet-File
Testablauf	Wie in den jeweiligen Tests.

Erwartetes Resultat:	Wie in den jeweiligen Tests.

Test ID: 8	
Name	Ändern des TsNet-File und der Drop-Down
Testvoraussetzungen	Ein Projekt mit mindestens zwei TsNet-Files
Testablauf	1. TsNet-File laden 2. Drop-Down betrachten 3. Anderes TsNet-File laden 4. Drop-Down betrachten
Erwartetes Resultat:	Die Drop-Down passt den Inhalte den Dateien entsprechend an.

Test ID: 9	
Name	Datei entfernt vor Klick auf RUN
Testvoraussetzungen	Vollständiges Projekt
Testablauf	1. Projekt laden 2. Control-Model, TsNet-File und Sheet auswählen 3. Entweder Control-Model oder TsNEt-Datei löschen 4. Klick auf „RUN“
Erwartetes Resultat:	Wenn eine Datei fehlt, sollte die Simulation nicht durchgeführt werden können. Eine entsprechende Meldung weist darauf hin.

9.3.2 Testergebnis

Testdatum: 14. April 2015
Testperson: Dominik Zgraggen

Test ID:	Erwartung Erfüllt?	Kommentar	Weitere Schritte

1	NEIN	Load-Button ist fälschlicherweise aktiviert	Fehler behoben:
			`% disable Load button, cause drp_LoadProj has default-Value`
			`if isempty(getDrpSelItem( ...`
			`  handles.drp_SelProj, ...`
			`  GuiConstants.DefProjSel))`
			`set(handles.btn_LoadProj,...`
			`  'Enable', 'off');`
			`end`

2	Ja	Meldung erscheint: "This Selection is not a valid Project. It has not got the Project-Folder-Structure"	

3	NEIN	Fehlermeldung erscheint nicht. Nach erneutem Laden stimmt alles wieder.	String'war im catch-Teil vergessen gegangen. try open_system(get(... handles.txt_CtrMdlPath, ... 'String')); catch msgbox([GuiConstants.ErrOpnFile ... get(handles.txt_CtrMdlPath, ... 'String')], ... GuiConstants.ErrOpnFileTitle); end
4	Ja	The selected File needs to be a Simulink-Model-File. / The selected File needs to be a Excel-File.	
5	NEIN	Bei leerem Control-Model: 'Could not Load MDL-File' und bei leerem TsNet-File: 'Simulation failed!'	Es wird eine Funktion implementiert, die folgendes überprüft: - Existiert Excel-Sheet: ‚Config'? - Wenn ja, ist der Wert der Zelle A1 = Application Test? - Wenn ja, wird ein richtiges TsNet-verwendet.
6	Ja		
7	Ja	Bei Testfall 3 gleicher Fehler, selbe Korrektur	
8	Ja		
9	NEIN	Wenn die Simulation trotz gelöschtem Control-Model gestartet wird, erscheint der Status: 'Could not load Control-Model' Falls die TsNet-Datei fehlt, erscheint die Meldung: 'TsNet: Read Error'	Code hinzugefügt: % check if the given Files have been delete if not(exist(get(... handles.txt_CtrMdlPath,'String'...))) || not(exist(get(... handles.txt_TsNetPath,'String'))) Run_state = 0; msgbox(GuiConstants.FileNotExist,... GuiConstants.FailRunTitle); end

Tabelle 9-1: Ergebnis White-Box-Test

9.3.3 Nachtest

Testdatum: 15. April 2015
Testperson: Dominik Zgraggen

Test ID:	Erwartung Erfüllt?	Kommentar	Weitere Schritte
1	JA	Load ist bei ‚Select existing Test-Project...' deaktiviert	
3	JA	Hinweis an Benutzer erscheint	
5	JA	„This Excel-File has no valid TsNet-Structure"	
9	JA	„A file doesn not exist anymore"	

Tabelle 9-2: Nachtest

9.4 Akzeptanztest

9.4.1 Testfälle

Test ID: 0	
Name	Startbedingung
Testvoraussetzungen	GUI wird korrekt geöffnet
Testablauf	-
Erwartetes Resultat:	Beim Öffnen des GUIs ist nur das Drop-Down-Menu für die Auswahl eines bestehenden Projekts, der „Create Project“-Button und die Hilfe-Buttons aktiviert.

Test ID: 1.1	
Name	Test-Project-Selection: Projektnamen in Drop-Down
Testvoraussetzungen	Startbedingungen
Testablauf	1. Drop-Down-Menu ausklappen 2. Projekt auswählen
Erwartetes Resultat:	Alle Projekte werden angezeigt und sind auswählbar. Nach einer Auswahl ist der Button „Load“ aktiviert.

Test ID: 1.2	
Name	Test-Project-Selection: Projekt laden
Testvoraussetzungen	Im Drop-Down-Menu wurde ein Projektname ausgewählt.
Testablauf	1. Auswahl im Drop-Down-Menu 2. Klick auf „Load“
Erwartetes Resultat:	- Your-Work-Path wird vollständig und die Auswahl im Drop-Down-Menu entsprechend angezeigt. - Info-Box für Projekte wechselt auf "OK" (grün) - Die Buttons „Select Application for Import“, „Select existing Control-Model“ und „Select TsNet-File“ werden aktiviert - Es sind nun alle Buttons ausser Load in "1. Test-Project-Selection" aktiviert (neu “Open Folder”, “Delete Project”, “Save Project As”) - Die vorhandenen Daten im Projekt werden in den Textfeldern Control-Model-Path oder TsNet-File-Path angezeigt (muss genau eine Datei sein, damit geladen wird). Die Info-Boxen und Modify-Buttons werden dementsprechend geändert. Wenn das TsNet-File geladen wurde, ist das Drop-Down-Menu zur Auswahl des Excel-Sheets aktiviert und weist den entsprechenden Inhalt auf.

Test ID: 1.3	
Name	Test-Project-Selection: Änderung des Projekts
Testvoraussetzungen	Projekt ausgewählt und geladen
Testablauf	1. Projekt auswählen und laden 2. Im Drop-Down-Menu für Projekt die Auswahl ändern
Erwartetes Resultat:	Buttons "Delete Project", "Save Project As" und "Open Folder" werden deaktiviert. Ausserdem werden die Info-Boxen auf Rot gesetzt. Die Buttons „Select Application for Import“, „Select existing Control-Model“, „Select TsNet-File“, „Modify Control-Model“, „Modify TsNet-File“ und das Drop-Down-Menu für die Excel-Sheet-Auswahl sind nun allesamt deaktiviert.

Test ID: 2.1	
Name	Control-Model Generation or Selection: bestehendes Control-Model
Testvoraussetzungen	Projekt ausgewählt
Testablauf	1. Klick auf den Button „Select existing Control-Model“ 2. Auswählen einer MDL-Datei
Erwartetes Resultat:	Es wird ein Fenster, dass nur MDL-Dateien auswählen lässt geöffnet. Nach der Auswahl einer Datei: - Info-Box des Control-Models wechselt auf "OK" (grün) - Der Pfad der MDL-Datei wird in Control-Model-Path sichtbar. (Falls die Datei ausserhalb des Projekts war, wird sie in den Unterordner kopiert) - Der Button "Modify Control-Model" wird aktiviert

Test ID: 2.2	
Name	Control-Model Generation or Selection: Control-Model bearbeiten
Testvoraussetzungen	Projekt und Control-Model ausgewählt
Testablauf	1. MDL-Datei gewählt 2. Klick auf „Modify Control-Model“ 3. Bearbeiten in Simulink
Erwartetes Resultat:	Der Klick öffnet das Model in Simulink.

Test ID: 3.1	
Name	TsNet-File and Excel-Sheet: TsNet-File auswählen
Testvoraussetzungen	Projekt ausgewählt
Testablauf	1. Klick auf den Button „Select TsNet-File“ 2. Auswahl eines Excel-Files (*.xls oder *.xlsx)

Erwartetes Resultat:	Es wird ein Fenster geöffnet, das Excel-Dateien anwählen lässt. Nach der Auswahl einer Datei: - Der Pfad der Excel-Datei wird in TsNet-File-Path sichtbar. (Falls die Datei ausserhalb des Projekts war, wird sie in den Unterordner kopiert) - Drop-Down-Menu "Select the Excel-Sheet…" enthält nun die Sheets und ist wie auch der Button "Modify TsNet-File" aktiviert worden.

Test ID: 3.2	
Name	TsNet-File and Excel-Sheet: Excel-Sheet auswählen
Testvoraussetzungen	Projekt und TsNet-File ausgewählt
Testablauf	1. TsNet-File ausgewählt 2. Excel-Sheet in Drop-Down auswählen
Erwartetes Resultat:	Info-Box für TsNet wechselt auf „OK“ (grün)

Test ID: 3.3	
Name	TsNet-File and Excel-Sheet: TsNet-File bearbeiten
Testvoraussetzungen	Projekt ausgewählt und TsNet-File gewählt
Testablauf	1. TsNet-File ausgewählt 2. Klick auf „Modify TsNet-File“ 3. Bearbeiten im Excel
Erwartetes Resultat:	Der Klick öffnet die entsprechende Datei im Excel.

Test ID: 4.1	
Name	Simulation and Evaluation: RUN aktiviert
Testvoraussetzungen	- Control-Model gewählt - TsNet-File gewählt - TsNet-Excel-Sheet gewählt
Erwartetes Resultat:	Wenn Control-Model, TsNet-File und TsNet-Sheet ausgewählt sind, wird der Button RUN aktiviert. Falls RUN geklickt werden konnte, ohne dass alle Voraussetzung erfüllt sind, wird der Benutzer darauf hingewiesen, ohne dass die Simulation gestartet wird.

Test ID: 4.2	
Name	Simulation and Evaluation: Simulation durchführen
Testvoraussetzungen	RUN ist aktiviert

Testablauf	1. Klick auf RUN
Erwartetes Resultat:	Simulation wird durchgeführt: - Es werden Statusmeldungen gezeigt Bei erfolgreicher Simulation: - Die Info-Box wechselt, wenn die Simulation vollständig durchlaufen wurde auf "OK" und (grün) - Button „Show Report“ aktivieren Bei fehlgeschlagener Simulation: - Im Status-Textfeld steht was die Ursache der fehlgeschlagenen Simulation ist. Der Report kann nicht angesehen werden.

Test ID: 4.3	
Name	Simulation and Evaluation: Report ansehen
Testvoraussetzungen	Simulation erfolgreich durchgeführt
Testablauf	1. Simulation wird nach Klick auf „RUN“ mit positiver Rückmeldung abgeschlossen 2. Klick auf Button „Show Report“
Erwartetes Resultat:	Das TsNet-File wird geöffnet mit dem Report als Excel-Sheet.

Test ID: 5.1	
Name	GUI-Gesamt: Textfelder eingeschränkt
Testablauf	1. Klick in Textfeld 2. Eintippen auf Tastatur
Erwartetes Resultat:	Alle Textfelder sollten keine Texteingabe zulassen.

Test ID: 5.2	
Name	GUI-Gesamt: Help-Texte
Testablauf	1. Klick auf die Help-Buttons 2. Vergleichen der Texte
Erwartetes Resultat:	Jeder Button "?" oder "Help" zeigt verschiedene Hilfetexte.

9.4.2 Testergebnis

Testdatum: 15. April 2015
Testperson: Toni Kryenbuehl

Testfall-ID:	Erwartung Erfüllt?	Kommentar	Weitere Schritte
0	Ja		
1.1	Ja		
1.2	Ja	Mit einem und mehreren Control Model	
1.3	Ja		
2.1	Ja	Pfad gewechselt	
2.2	Ja		
3.1	Ja		
3.2	Ja		
3.3	Ja		
4.1	Ja		
4.2	Ja		
4.3	Ja		
5.1	Ja		
5.2	Ja		
	Ja		Show Report während RUN -> kein Abrsturz
	Ja	Fehlermeldung nicht möglich	Select TsNet mit anderem Dateityp -> kein Abrsturz
	Ja	Auswahl möglich	Select existing Control Model mit anderem Dateityp (TSNetVAV2.mdl.autosave) -> kein Abrsturz
	Ja	Zweckmässig	Übersichtlichkeit GUI
	Ja	Zweckmässig	Übersichtlichkeit Help-Texte

Tabelle 9-3: Ergebnis Akzeptanztest

10. Schlusswort

Bereits vor Beginn der IPA habe ich mich intensiv mit IMSES und dem Open-Loop-Testing befasst. Dass ich den detaillierten Workflow verstanden habe, half mir während der gesamten GUI-Entwicklung. Nach der Analyse der Use Cases im Vorfeld empfand ich die Aufgabenstellung meiner IPA nicht mehr als besonders schwierig. Ich profitierte von einem effektiven Zeitplan und einer sinnvollen Projektmanagement-Methode. Die eingeplante Pufferzeit bewirkte, dass ich nie unter Zeitdruck geriet. Beim Dokumentieren stellte sich immer die Frage nach der Menge. Ich versuchte einfach alles, was getan und überlegt wurde, festzuhalten. Nur so kann die Arbeit richtig bewertet werden.

Das Design des GUIs ermöglichte auf den ersten Blick grosse Freiheiten. Wenn ich mich in den Benutzer hineinversetzte, musste ich jedoch meistens das Gegenteil feststellen. Ein GUI zweckmässig zu entwerfen, ist aufwändige Denkarbeit. Beim Code schreiben in der IPA konnte ich meine MATLAB-Kenntnisse deutlich verbessern. Mein Code ist mit Sicherheit nicht optimal geschrieben, da ich meinen Programmierstil dem Zeitdruck anpasste. Ich denke, dass ich die Aufgabe, meinen Code mit dem Bestehenden zu verbinden, gut gelöst habe. Im bestehenden Code waren schlussendlich nur wenige Anpassungen nötig. Ich hätte nicht gedacht, dass ich in dieser relativ kurzen Zeit, zirka 800 Zeilen Code schreiben werde. Das Testen empfand ich als Herausforderung, da ich weder von der Schule noch vom Lehrbetrieb nennenswerte Fachkompetenzen mitbrachte. Ich denke, dass sich die Testfälle mit gesundem Menschenverstand einfach ermitteln und ausführen lassen. Deshalb schätze ich meine Qualitätssicherung als genügend ein.

Am Ende meines dritten Lehrjahres war ich unsicher, ob ich erfolgreich ein GUI entwerfen kann. Aus diesem Grund wollte ich es im letzten Lehrjahr versuchen. Meine Fertigkeit etwas optisch schön zu gestalten hat deutliches Verbesserungspotenzial. Das GUI ergonomisch zu kreiern, war meiner Meinung nach wichtiger in meiner Aufgabe. Ich bin mit dem Ergebnis zufrieden und blicke auf tolle zehn Tage zurück. Ich kann mir gut vorstellen, nach Lehre und Studium in diesem Bereich eine Stelle zu suchen.

11. Anhang

11.1 Workflow-Analyse (vor IPA)

11.1.1 Flussdiagramm

„Auf nächster Seite“

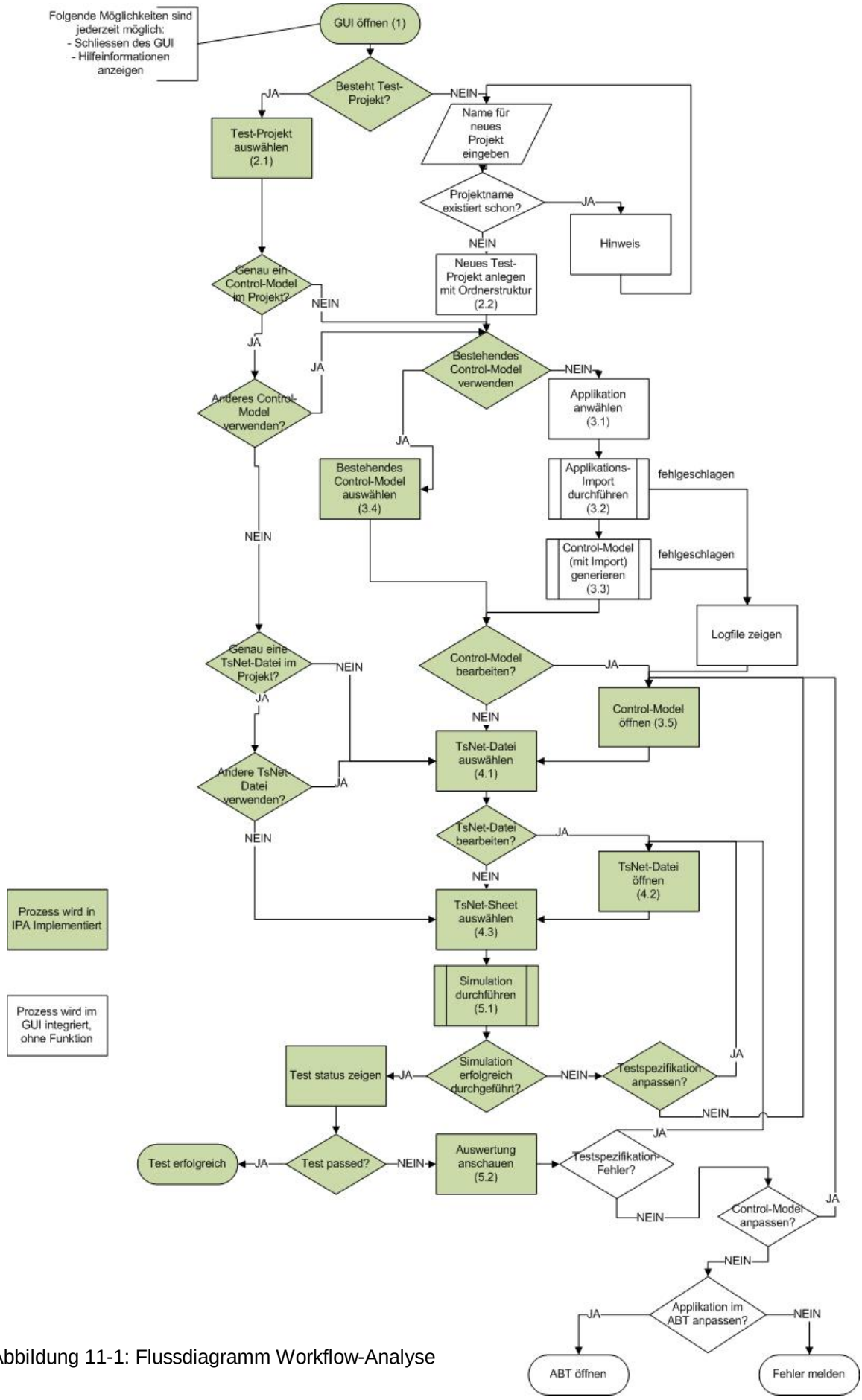


Abbildung 11-1: Flussdiagramm Workflow-Analyse

11.1.2 Beschreibung Flussdiagramm

Name und Inhalt	
Eingangsdaten	*(externe Daten, IMSES-Intere Daten, Settings, Pfade)*
Eingangsvoraussetzungen	*(Schritte, die vorher gemacht sein müssen und deren Gesamtergebnis)*
Benutzereingaben	*(wenn erforderlich)*
Stati: Warnungen oder Fehler	
Ausgangsdaten	*(nach extern, IMSES-intern)*
Gesamtergebnis	*(OK /nicht OK / Warnung)*

Triviale Prozesse werden bewusst weggelassen.

Name und Inhalt	GUI öffnen (1): Run-Befehl der Matlab-Datei
Eingangsdaten	Stammverzeichnis des GUIs
Eingangsvoraussetzungen	falls kein Unterordner "Projects" vorhanden ist, wird dieser erstellt
Benutzereingaben	-
Stati: Warnungen oder Fehler	Matlab-Datei kann nicht ausgeführt werden. (Fehler) hat nicht nötige Berechtigung um Ordner anzulegen. (Warnung)
Ausgangsdaten	Falls noch nicht vorhanden: Erstellter Unterordner "Projects"
Gesamtergebnis	-

Name und Inhalt	Test-Projekt auswählen (2.1): Ein früher angelegtes Projekt besteht bereits. Der Projekt-Ordner wird angewählt. Ein Projekt ist ein Windows-Verzeichnis innerhalb von "Projects", mit vordefinierter Unterordner-Struktur, wo Test-Daten, wie TsNet-Datei, Modelle usw. abgelegt sind.
Eingangsdaten	-
Eingangsvoraussetzungen	Es würde einmal ein Test-Projekt angelegt, welches im Unterordner "Projects" befindet und die vorgegebene Pojektordner-Struktur enthält.
Benutzereingaben	Windows-Ordner innerhalb von "Projects"
Stati: Warnungen oder Fehler	Ordner nicht innerhalb von Projects (Warnung) Projekt-Ordner hat nicht vordefinierte Struktur (Fehler)
Ausgangsdaten	Die Test-Daten, welche im ausgewählten Test-Projekt vorhanden waren, werden im GUI sichtbar gemacht. wenn genau eine Datei dafür in Frage kommt. (Bei keiner oder mehr als einer, muss der User später manuell auswählen)
Gesamtergebnis	Test-Projekt ist dem GUI bekannt und die vorhandenen Dateien sind im GUI sichtbar.

Name und Inhalt	Neues Test-Projekt anlegen mit Ordnerstruktur (2.2): Der erzeugte Ordner im Unterordner "Projects" trägt den eingegeben Namen und enthält die vordefinierte Ordnerstruktur.

Eingangsdaten	Projektname
Eingangsvoraussetzungen	Projektname ist eingegeben
Benutzereingaben	-
Stati: Warnungen oder Fehler	Projektname existiert bereits (Warnung, anderer Name erzwingen / Projekt nicht erstellen)
Ausgangsdaten	Projektverzeichnis angelegt
Gesamtergebnis	GUI kennt das "working"-Projekt, welches vollständig angelegt wurde.

Name und Inhalt	Applikation anwählen (3.1): Der ABT-Export (Applikation) in Form einer Zip-Datei wird für den Import festgelegt.
Eingangsdaten	-
Eingangsvoraussetzungen	Test-Projekt ausgewählt
Benutzereingaben	Ausgewählte Zip-Datei wird in den entsprechenden Projekt-Ordner kopiert.
Stati: Warnungen oder Fehler	Datei ist kein Zip (Fehler, kann verunmöglicht werden)
Ausgangsdaten	Pfad der Applikation (Zip-Datei)
Gesamtergebnis	Applikation wird in entsprechenden Ordner kopiert und wird der Pfad wird im GUI sichtbar.

Name und Inhalt	Applikations-Import durchführen (3.2): Aus der Applikation (Zip-Datei) wird mit dem IMSES-Importer in eine Model-Datei erzeugt.
Eingangsdaten	Applikation (Zip-Datei)
Eingangsvoraussetzungen	Test-Projekt ausgewählt und Zip-Datei existiert im entsprechenden Unterordner
Benutzereingaben	-
Stati: Warnungen oder Fehler	Applikation enthält schwerwiegende Fehler und kann nicht korrekt Importiert werden (Fehler, Logfile gibt Aufschluss über weiteres Vorgehen: Control-Model anpassen?)
Ausgangsdaten	Entsprechende Meldungen in Logfile. Model-Datei, welche der IMSES-Importer erzeugt hat entsprechend abgelegt.
Gesamtergebnis	Logfile enthält notwendigen Informationen und Model-Datei abgelegt.

Name und Inhalt	Control-Model (mit Import) generieren (3.3): Mit dem vorher importierten Applikations-Model wird das Control-Model generiert (mit Hilfe von XFB-Blöcke "BA_Cominterf" und "BA_Cominterf_Out").
Eingangsdaten	Applikations-Model-Datei im entsprechenden Unterordner
Eingangsvoraussetzungen	
Benutzereingaben	-

Stati: Warnungen oder Fehler	Probleme beim Erstellen des Control-Models (Fehler, Logfile gibt Aufschluss über weiteres Vorgehen: Control-Model anpassen?)
Ausgangsdaten	Control-Model (.MDL-Datei) im entsprechenden Unterordner
Gesamtergebnis	Aus der Applikation wird ein Control-Model, welches später so in der Simulation verwendet werden kann.

Name und Inhalt	Bestehendes Control-Model auswählen (3.4): Der User sollte eine Model-Datei auswählen. Es sollte früher einmal von IMSES generiert worden sein, d.h. die Applikation ist als "Control" darin und die XFB-Blöcke "BA_Cominterf" und "BA_Cominterf_Out" sind vorhanden.
Eingangsdaten	
Eingangsvoraussetzungen	Test-Projekt ausgewählt
Benutzereingaben	Control-Model (.mdl-Datei) vom Benutzer angewählt. Falls sich dieses nicht innerhalb des Test-Projekts befindet, wird es in den entsprechenden Unterordner kopiert.
Stati: Warnungen oder Fehler	Control-Model kann nicht in entsprechenden Ordner kopiert werden (Fehler) ausgewählte Datei hat falschen Dateityp (Fehler, kann verunmöglicht werden)
Ausgangsdaten	Pfad des Control-Models wird auf dem GUI sichtbar gemacht.
Gesamtergebnis	User definiert, die Control-Model-Datei und dies ist im GUI ersichtlich.

Name und Inhalt	Control-Model öffnen (3.5): Das Model im Simulink anzeigen, damit es bearbeitet werden kann.
Eingangsdaten	Der Pfad des Control-Models
Eingangsvoraussetzungen	Test-Projekt ausgewählt und es ist bereits ein bestehendes Control-Model ausgewählt (im GUI sichtbar).
Benutzereingaben	-
Stati: Warnungen oder Fehler	MDL-Datei kann nicht geöffnet werden (Fehler) angepasste Datei kann nicht gespeichert werden (Fehler)
Ausgangsdaten	bearbeitetes Control-Model
Gesamtergebnis	Control-Model wurde im Simulink geöffnet, bearbeitet und gespeichert.

Name und Inhalt	TsNet-Datei auswählen (4.1): Der User sollte eine TsNet-Datei auswählen. Es beinhaltet die Testspezifikation.
Eingangsdaten	-
Eingangsvoraussetzungen	Test-Projekt ausgewählt und Control-Model sind im GUI sichtbar.
Benutzereingaben	TsNet-Datei (.xls- oder .xlsx-Datei), falls sich diese nicht innerhalb des Test-Projekts befindet, wird es in den entsprechenden Unterordner kopiert.

Stati: Warnungen oder Fehler	TsNet-Datei kann nicht in entsprechenden Ordner kopiert werden (Fehler) ausgewählte Datei hat falschen Dateityp (Fehler, kann verunmöglicht werden)
Ausgangsdaten	Pfad der TsNet-Datei wird auf dem GUI sichtbar gemacht.
Gesamtergebnis	Der Pfad der TsNet-Datei wurde vom User angewählt.

Name und Inhalt	TsNet-Datei öffnen (4.2): TsNet-Datei mit Excel anzeigen, damit es bearbeitet werden kann.
Eingangsdaten	Der Pfad der TsNet-Datei
Eingangsvoraussetzungen	Test-Projekt ausgewählt und Excel ist installiert.
Benutzereingaben	-
Stati: Warnungen oder Fehler	Excel-Datei kann nicht geöffnet werden (Fehler) angepasste Datei kann nicht gespeichert werden (Fehler)
Ausgangsdaten	bearbeitete TsNet-Datei
Gesamtergebnis	Die TsNEt-Datei wurde im Excel geöffnet, bearbeitet und gespeichert.

Name und Inhalt	TsNet-Sheet auswählen (4.3): Um ein Test durchzuführen wird nur ein Sheet aus der Excel-Datei benötigt. Welches es ist, muss vom User definiert werden.
Eingangsdaten	Auflistung aller Sheets (Tabellenblätter)
Eingangsvoraussetzungen	Test-Projekt ausgewählt Sheets können aus Excel-Datei ausgelesen werden.
Benutzereingaben	Sheet aus Liste auswählen
Stati: Warnungen oder Fehler	Sheets können nicht ausgelesen werden (Fehler)
Ausgangsdaten	Sheet-Name in GUI sichtbar
Gesamtergebnis	Dem GUI ist bekannt, welches Sheet für den Test benutzt wird.

Name und Inhalt	Simulation durchführen (5.1): Für den Aufruf der Matlab funktion "sim" werden die Parameter Control-Model und TsNet-Sheet benötigt.
Eingangsdaten	Control-Model, TsNet-Sheetname (mit Pfad, der Excel-Datei)
Eingangsvoraussetzungen	Test-Projekt ausgewählt und Eingangsdaten sind ausgewählt
Benutzereingaben	-
Stati: Warnungen oder Fehler	Simulation fehlgeschlagen (Fehler) Test fehlgeschlagen (Warnung) Nicht alle benötigten Parameter vorhanden (Warnung, kann verunmöglicht werden)
Ausgangsdaten	Excel-Report abgelegt im entsprechenden Ordner
Gesamtergebnis	Simulation durchgeführt: Status der Simulation und des Tests auf GUI sichtbar, Report als Excel-Datei abgelegt.

Name und Inhalt	Auswertung anschauen (5.2): User will Einsicht in die Testergebnisse. Er kann den Report, der in der Simulation erstellt wurde, im Excel anschauen.
Eingangsdaten	Excel-Report im entsprechenden Unterordner
Eingangsvoraussetzungen	Test-Projekt ausgewählt, Simulation durchgeführt (Report besteht)
Benutzereingaben	-
Stati: Warnungen oder Fehler	Report nicht vorhanden oder kann nicht geöffnet weren (Fehler)
Ausgangsdaten	-
Gesamtergebnis	User konnte die Einzelheiten aus dem Report entnehmen.

Name und Inhalt	Test erfolgreich: Voraussetzung für einen erfolgreichen Test ist einerseits die erfolgreiche Simulation (Control-Model und TsNet-Testspezifikation sind korrekt aufgebaut), andererseits dass der Report mit den Erwartungswerten übereinstimmt.
Eingangsdaten	Control-Model, TsNet-Sheetname (mit Pfad, der Excel-Datei)
Eingangsvoraussetzungen	Simulation wurde gestartet und ohne Fehler beendet.
Benutzereingaben	-
Stati: Warnungen oder Fehler	-
Ausgangsdaten	-
Gesamtergebnis	Open-Loop Test der Applikation ist abgeschlossen.

Tabelle 11-1: Flussdiagramm Beschreibung

11.2 CODE

11.2.1 IMSES_GUI.m (vollständig neu)

```
function varargout = IMSES_GUI(varargin)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   (C) Copyright by Siemens Schweiz AG, Building Technologies Group,
%       HVAC Products, 2015
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Project                         : IMSES
%   Target Hardware                 : PC
%   Target Operating System         : WinXP / Win7 Console
%   Language/Compiler               : Matlab 2010 and higher
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Workfile                        : IMSES_GUI.m
%   Author                          : Dominik Zgraggen
%   Version                         : v1.0
%   Date                            : 10-April-2015
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Matlab Informations
%  IMSES_GUI.m is code for IMSES_GUI.fig.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:
%   Run this file for open GUI.
%   the Subfolders: Projects, functions are needed.
%   Also GuiConstants.m is needed.
%   There are Two Parts:
%   - Callback-Functions (gets called from btn (=Button) or drp (=Dropdown)
%   - Additional-Functions (Code is used more than once
%      and/or less Code in Callback)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function/Interface:
%   INPUT: varargin   command line arguments to IMSES_GUI (see VARARGIN)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Revision History
%   (Put meaningful comments in SourceSafe for log below!)
%   (Please remove blank lines and very old comments!)
%
%   Document Creation (For IPA)
%   2015-04-10  Dominik Zgraggen, 5559
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @IMSES_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @IMSES_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before IMSES_GUI is made visible.
function IMSES_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to IMSES_GUI (see VARARGIN)

% Choose default command line output for IMSES_GUI
    handles.output = hObject;

    % struct contains Data such as Path (collects the Variable)
    handles.GuiProperties = struct( ...
        'BasePath', '',... % Matlab-File root
        'ProjectPath', '',... % Users ProjectPath
        'ProjectName', '',... % Name of Project (also Foldername)
        'Mdl_File', '',... % Users Control-Model - full filepath
        'SelectedSheet', '',... % name of Excel-Sheet in TsNet-File
        'TsNet_File', '');  % Users TsNet-File - full filepath

    % find BasePath - where is this Matlab-File?
    [pathstr] = fileparts(which('IMSES_GUI.m'));
    handles.GuiProperties.BasePath = pathstr;

    % find all Projects for Drowdown: drp_SelProj
    folders = dir([handles.GuiProperties.BasePath '\Projects']); % find
folderlist
    foldernames = {folders.name};
    i = 0;

    while i < length(foldernames)   % remove unexisting Folders
        if strfind(foldernames{i+1}, '.')   % there are folders like this
            foldernames{i+1} = [];
        end
        i = i + 1;
    end

    foldernames = foldernames(~cellfun('isempty',foldernames));
    arr = [GuiConstants.DefProjSel foldernames]; % add default
    set(handles.drp_SelProj, 'String', arr); % set Dropdown item

    % set Properties, Data stil there? (Matlab not closed)
    handles.GuiProperties.ProjectPath = get(handles.txt_WorkPath, 'String');
    handles.GuiProperties.Mdl_File = get(handles.txt_CtrMdlPath, 'String');
    handles.GuiProperties.TsNet_File = get(handles.txt_TsNetPath, 'String');

    % Dropdowns
    handles.GuiProperties.ProjectName = ...
    getDrpSelItem(handles.drp_SelProj, GuiConstants.DefProjSel);

    updateGUI(hObject, handles);

    % Update handles structure
    guidata(hObject, handles);
end

% --- Outputs from this function are returned to the command line.
```

```
function varargout = IMSES_GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

end

end % IMSES_GUI needs to be closed here


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%% CALLBACK-FUNCTIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functionname-convention: Type_Tag_Function()

%% PARAMETER %%
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% global Help shows Workflow-Information
function btn_help_Callback(hObject, eventdata, handles)
    msgbox(GuiConstants.HelpMes, GuiConstants.HelpMesTitle);
end

% shows Project specific Information
function btn_QProj_Callback(hObject, eventdata, handles)
    msgbox([GuiConstants.ProMes1 GuiConstants.ProMes2 ...
        GuiConstants.ProMes3 GuiConstants.ProMes4 GuiConstants.ProMes5],...
        GuiConstants.ProMesTitle);
end

% shows Control-Model specific Information
function btn_QCtrMdl_Callback(hObject, eventdata, handles)
    msgbox([GuiConstants.MdlMes1 GuiConstants.MdlMes2 ...
        GuiConstants.MdlMes3 GuiConstants.MdlMes4],...
        GuiConstants.MdlMesTitle);
end

% shows TsNet specific Information
function btn_QTsNet_Callback(hObject, eventdata, handles)
    msgbox([GuiConstants.TsnMes1 GuiConstants.TsnMes2 ...
        GuiConstants.TsnMes3], GuiConstants.TsnMesTitle);
end

% shows Simulation and Report specific Information
function btn_QSim_Callback(hObject, eventdata, handles)
    msgbox([GuiConstants.SimMes1 GuiConstants.SimMes2 ...
        GuiConstants.SimMes3], GuiConstants.SimMesTitle);
end

% btn_LoadProj_Callback is called ,when "Load"-Button was pressed
% loads files of the project, which is selected in drp_SelProj
function btn_LoadProj_Callback(hObject, eventdata, handles)
```

```
    %get string of the item
    items = get(handles.drp_SelProj,'String');
    drp_index = get(handles.drp_SelProj,'Value');
    drp_item = items{drp_index};
    path = [handles.GuiProperties.BasePath '\Projects\' drp_item];
    noPro = 0;

  if isdir(path) == 1
      % check if folder has Project structur
      if isdir([path '\' GuiConstants.ApplDir]) ~= 1
          noPro = 1;
      elseif isdir([path '\' GuiConstants.CtrMdlDir]) ~= 1
          noPro = 1;
      elseif isdir([path '\' GuiConstants.TsNetDir]) ~= 1
          noPro = 1;
      elseif isdir([path '\' GuiConstants.AddDir]) ~= 1
          noPro = 1;
      end
      if (noPro == 0)
        set(handles.txt_WorkPath, 'String', path)
        handles.GuiProperties.ProjectPath = path;
        handles.GuiProperties.ProjectName = drp_item;

        %Load files if there are any
        %list mdl files
        files = dir([path '\' GuiConstants.CtrMdlDir '\*.mdl']);
        if length(files) == 1   % load the file if there is exactly one
            handles.GuiProperties.Mdl_File = [path '\' ...
                    GuiConstants.CtrMdlDir '\' files(1).name];
            set(handles.txt_CtrMdlPath, 'String', ...
                    handles.GuiProperties.Mdl_File);
        else
            % not one file to be loaded
            set(handles.txt_CtrMdlPath, 'String', '');
            handles.GuiProperties.Mdl_File = '';
        end
        filesXLS = dir([path '\' GuiConstants.TsNetDir '\*.xls']);
        filesXLSX = dir([path '\' GuiConstants.TsNetDir '\*.xlsx']);
        files = [filesXLS filesXLSX];
        if length(files) == 1
            handles.GuiProperties.TsNet_File = [path '\' ...
                GuiConstants.TsNetDir '\' files(1).name];
            set(handles.txt_TsNetPath, 'String', ...
                handles.GuiProperties.TsNet_File);

            % load Sheets
            [status,sheets]=xlsfinfo(handles.GuiProperties.TsNet_File);
            sheetsWithDef = [GuiConstants.DefSheetSel sheets];
            set(handles.drp_SelSheet,'String', sheetsWithDef);
        else
            handles.GuiProperties.TsNet_File = '';
            set(handles.txt_TsNetPath, 'String', '');
        end
      else
          % no valid Project structure
         msgbox(GuiConstants.ProjStruc, GuiConstants.ProjStrucTitle);
      end
  end

  updateGUI(hObject, handles);
```

```
  guidata(hObject, handles);
end

% btn_CreateProj_Callback force input for Projectname
function btn_CreateProj_Callback(hObject, eventdata, handles)
    ProjectName = inputdlg();   % not Part of IPA
end

% btn_OpnProj_Callback is called, when Folder should be shown in Explorer
function btn_OpnProj_Callback(hObject, eventdata, handles)
    try
        system(['explorer.exe ' get(handles.txt_WorkPath, 'String')]);
    catch
        Msgbox([GuiConstants.OpnProj get(handles.txt_WorkPath)], ...
            GuiConstants.OpnProjTitle);
    end
end

% btn_SelCtrMdl_Callback is called, when a Control-Model should be selected
function btn_SelCtrMdl_Callback(hObject, eventdata, handles)
    if not(isempty(handles.GuiProperties.ProjectName))
        path = [handles.GuiProperties.BasePath '\Projects\' ...
         handles.GuiProperties.ProjectName '\' GuiConstants.CtrMdlDir '\'];
        [filename,pathname]=uigetfile('*.mdl',...
            GuiConstants.CtrMdlSel, path);


        % if Selected Model is not in Project, copy it there
        if filename ~= 0
            if strfind(filename, '.mdl')
                if not(strcmp(pathname, path))
                    try
                        copyfile([pathname '\' filename], [path filename])
                        handles.GuiProperties.Mdl_File=[path filename];
                        set(handles.txt_CtrMdlPath,'String', ...
                            [path filename]);
                    catch
                        messagebox(['The File' pathname '\' filename ...
                            ' could not be copied to' path filename]);
                        handles.GuiProperties.Mdl_File='';
                        set(handles.txt_CtrMdlPath,'String', '');
                    end
                else
                    %selected Model is already in /Project/Control_Model
                    handles.GuiProperties.Mdl_File=[path filename];
                    set(handles.txt_CtrMdlPath,'String', [path filename]);
                end
            else
                % selected File is not MDL-File
                msgbox(GuiConstants.NoMdl, GuiConstants.NoMdlTitle);
            end
        end
    end
    updateGUI(hObject, handles);
    guidata(hObject, handles);
end

% btn_ModiCtrMdl_Callback is called, when Control-Model should be modified
function btn_ModiCtrMdl_Callback(hObject, eventdata, handles)
    try
```

```
        open_system(get(handles.txt_CtrMdlPath, 'String'));
    catch
        msgbox([GuiConstants.ErrOpnFile get(handles.txt_CtrMdlPath, ...
            'String')], GuiConstants.ErrOpnFileTitle);
    end
end

% btn_SelTsNet_Callback is called, when a TsNEt-File should be selected
function btn_SelTsNet_Callback(hObject, eventdata, handles)
    if not(isempty(handles.GuiProperties.ProjectName))
        path = [handles.GuiProperties.BasePath '\Projects\' ...
          handles.GuiProperties.ProjectName '\' GuiConstants.TsNetDir '\'];
        [filename,pathname]=uigetfile('*.xls;*xlsx', ...
                GuiConstants.TsNetSel, path);

        % if Selected File is not in Project, copy it there
        if filename ~= 0
            if strfind(filename, '.xls')
                if not(strcmp(pathname, path))
                    try
                        copyfile([pathname '\' filename], [path filename]);
                        handles.GuiProperties.TsNet_File=[path filename];
                        set(handles.txt_TsNetPath,'String', ...
                            [path filename]);
                    catch
                        messagebox(['The File' pathname '\' filename ...
                            ' could not be copied to' path filename]);
                        handles.GuiProperties.TsNet_File='';
                        set(handles.txt_TsNetPath,'String', '');
                    end
                else
                    %selected TsNe-File is already in /Project/TsNet
                    handles.GuiProperties.TsNet_File=[path filename];
                    set(handles.txt_TsNetPath,'String', [path filename]);
                end
                %load the Sheets/values for drp_SelSheet
                if not(isempty(handles.GuiProperties.TsNet_File))
                    [status,sheets]=xlsfinfo...
                        (handles.GuiProperties.TsNet_File);
                    sheetsWithDef = [GuiConstants.DefSheetSel sheets];
                    set(handles.drp_SelSheet,'String', sheetsWithDef);
                end
            else
                % selected File is no Excel-File
                msgbox(GuiConstants.NoXls, GuiConstants.NoXlsTitle);
            end
        end
    end
    updateGUI(hObject, handles);
    guidata(hObject, handles);
end

% btn_ModiTsNet_Callback is called, when the TsNet-File should modified
function btn_ModiTsNet_Callback(hObject, eventdata, handles)
    try
        winopen(get(handles.txt_TsNetPath, 'String'));
    catch
        msgbox([GuiConstants.ErrOpnFile get(handles.txt_TsNetPath, ...
            'String')], GuiConstants.ErrOpnFileTitle);
    end
```

```
end

% btn_RUN_Callback is called, when the user want to start the Simulation
function btn_RUN_Callback(hObject, eventdata, handles)
    % is there a Excel and a Model?
    Run_state = 1;  % 0 = don't run, 1 = parameter are ok

    if isempty(handles.GuiProperties.SelectedSheet) || ...
      strcmp(handles.GuiProperties.SelectedSheet, GuiConstants.DefSheetSel)
        Run_state = 0;  % 0 = sheet is not set/selected
        msgbox([GuiConstants.FailRun 'Excel-Sheet'], ...
            GuiConstants.FailRunTitle);
    end

    if not(strfind(get(handles.txt_TsNetPath,'String'), '.xls'))
        Run_state = 0;  % 0 = Parameter is no valid Excel-File
        msgbox([GuiConstants.FailRun 'TsNet-File'], ...
            GuiConstants.FailRunTitle);
    end

    if not(strfind(get(handles.txt_CtrMdlPath,'String'), '.mdl'))
        Run_state = 0;  % 0 = Parameter is no valid Model
        msgbox([GuiConstants.FailRun 'Control-Model'], ...
            GuiConstants.FailRunTitle);
    end

    if isXlsTsNet(hObject, handles) ~= 0
        Run_state = 0; % Excel-File is not correct TsNet
        msgbox(GuiConstants.XlsNoTsn, GuiConstants.FailRunTitle);
    end

     % check if the given Files have been delete
    if not(exist(get(handles.txt_CtrMdlPath,'String'))) || ...
        not(exist(get(handles.txt_TsNetPath,'String')))
        Run_state = 0;
        msgbox(GuiConstants.FileNotExist, GuiConstants.FailRunTitle);
    end

    if Run_state == 1       % 1 means Parameters are ok
            % Starts Simulation (FOREIGN CODE)
            TSNet_Test(get(handles.txt_TsNetPath,'String'),char(...
                handles.GuiProperties.SelectedSheet),get( ...
                handles.txt_CtrMdlPath,'String'), handles);
    else
            set(handles.txt_State, 'String', '...')
    end

    guidata(hObject, handles);
    updateGUI(hObject, handles);
end

% btn_OpnReport_Callback is called, when the Report-Sheet should be shown
function btn_OpnReport_Callback(hObject, eventdata, handles)
    try
        winopen(get(handles.txt_TsNetPath, 'String'));  % open Excel
    catch
        Msgbox([GuiConstants.ErrOpnFile get(handles.txt_TsNetPath)], ...
            GuiConstants.ErrOpnFileTitle);
    end
```

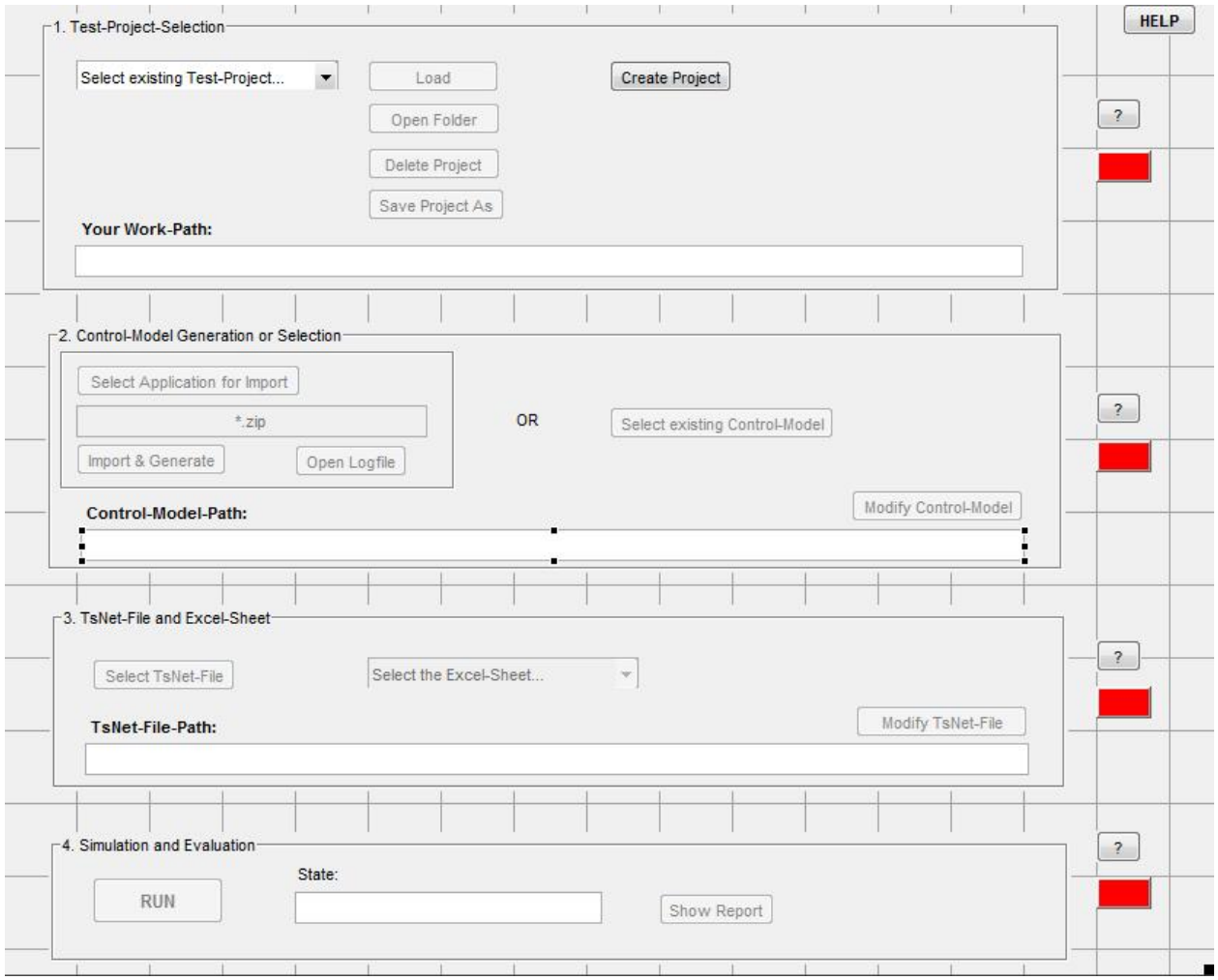
```
end

% drp_SelProj_Callback is called, when drp_SelProj has been changed
function drp_SelProj_Callback(hObject, eventdata, handles)
    updateGUI(hObject, handles);  %enables Load-Button
    guidata(hObject, handles);
end

% drp_SelSheet_Callback is called, when drp_SelSheet has been changed
% it sets the Property: SelectedSheet and calls updateGUI
function drp_SelSheet_Callback(hObject, eventdata, handles)
    if not(isempty(get(handles.txt_TsNetPath, 'String')))
        items = get(handles.drp_SelSheet,'String');
        drp_index = get(handles.drp_SelSheet,'Value');
        drp_item = items{drp_index};
        if not(strcmp(drp_item, GuiConstants.DefSheetSel))
            handles.GuiProperties.SelectedSheet = drp_item;
        end
    end

    updateGUI(hObject, handles)
    guidata(hObject, handles);
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% ADDITIONAL-FUNCTIONS %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Parameter %%
% hObject    handle to figure (needed fur guidata())
% handles    structure with handles and user data (see GUIDATA)

% updateGUI is called after every GUI-interaction
% it sets enable property of buttons and dropdowns, InfoBoxes
function updateGUI(hObject, handles)
    TextVal = get(handles.txt_WorkPath,'String');

    %get selected item of drp_SelProj
    drp_item = getDrpSelItem(handles.drp_SelProj, GuiConstants.DefProjSel);

    % is WorkPath not set?
    if isempty(TextVal)
        % Project hasn't been selected -> set InfoBoxes red
        falseInfoBoxes(hObject, handles);
        %disable all buttons, (except Load) -> there's no project
        disableButtons(hObject, handles);

        if isempty(drp_item)
            % Project has NOT been selected in dropdown
            set(handles.btn_LoadProj, 'Enable', 'off');
        else
            %Project has been selected in dropdown -> enable Loadbtn
            set(handles.btn_LoadProj, 'Enable', 'on');
        end
    else
        % Project Path is set
        % check if Dropdown-Selection is same Project as txt_WorkPath
        if strcmp([handles.GuiProperties.BasePath ...
```

```
            '\Projects\' drp_item], TextVal)
            afterPathIsSet(hObject, handles);

            % disable Load button, cause Project is loaded now
            set(handles.btn_LoadProj, 'Enable', 'off');

            % check CtrMdl set?
            if isempty(get(handles.txt_CtrMdlPath, 'String'))
                set(handles.box_CtrMdl, 'BackgroundColor', 'red');
                set(handles.box_CtrMdl, 'String', '');
            else
                set(handles.box_CtrMdl, 'BackgroundColor', 'green');
                set(handles.box_CtrMdl, 'String', 'OK');
                set(handles.btn_ModiCtrMdl, 'Enable', 'on');
            end

            % check TsNet-File set?
            if isempty(get(handles.txt_TsNetPath, 'String'))
                set(handles.box_TsNet, 'BackgroundColor', 'red');
                set(handles.box_TsNet, 'String', '');
                set(handles.btn_ModiTsNet, 'Enable', 'off');
                set(handles.drp_SelSheet, 'Enable', 'off');
            else
                %TsNet-File is set
                set(handles.btn_ModiTsNet, 'Enable', 'on');
                % enable Dropdown SelSheet
                set(handles.drp_SelSheet, 'Enable', 'on');
                % is dropdown selected?
                if not(isempty(getDrpSelItem(handles.drp_SelSheet, ...
                        GuiConstants.DefSheetSel)))
                    set(handles.box_TsNet, 'BackgroundColor', 'green');
                    set(handles.box_TsNet, 'String', 'OK');
                else
                    set(handles.box_TsNet, 'BackgroundColor', 'red');
                    set(handles.box_TsNet, 'String', '');
                end
            end
            checkRunEnable(hObject, handles);
        else
            % Project was set, but selProj-Dropdown has been changed
            % ..block user from doing further actions= not sure which Proj
            falseInfoBoxes(hObject, handles);
            disableButtons(hObject, handles);
            set(handles.btn_LoadProj, 'Enable', 'on');

            % disable Load button, cause drp_LoadProj has default-Value
            if isempty(getDrpSelItem(handles.drp_SelProj, ...
                    GuiConstants.DefProjSel))
                set(handles.btn_LoadProj, 'Enable', 'off');
            end
        end
    end
    guidata(hObject, handles);
end

% afterPathIsSet enables buttons, gets called when the Project path is set
function afterPathIsSet(hObject, handles)
    set(handles.box_Proj, 'BackgroundColor', 'green');
    set(handles.box_Proj, 'String', 'OK');
    set(handles.btn_OpnProj, 'Enable', 'on');
```

```
    set(handles.btn_DelProj, 'Enable', 'on');
    set(handles.btn_SavProjAs, 'Enable', 'on');
    set(handles.btn_SelAppl, 'Enable', 'on');
    set(handles.btn_SelCtrMdl, 'Enable', 'on');
    set(handles.btn_SelTsNet, 'Enable', 'on');
    guidata(hObject, handles);
end

% checkRunEnable checks RUN-Button enable Property
% missing inputs enable = off, all inputs done = on
function checkRunEnable(hObject, handles)
    enab = 1;       % 1 means disable RUN-Button
    if not(isempty(get(handles.txt_CtrMdlPath, 'String')))
        if not(isempty(get(handles.txt_TsNetPath, 'String')))
            if not(isempty(getDrpSelItem(handles.drp_SelSheet, ...
                    GuiConstants.DefSheetSel)))
                enab = 0;
            end
        end
    end

    if enab == 0
        %all inputs done, RUN enable
        set(handles.btn_RUN, 'Enable', 'on');
    else
        set(handles.btn_RUN, 'Enable', 'off');
    end

    % enable "show Report"-Button, set infoBox-Sim
    if strcmp(get(handles.txt_State, 'String'), GuiConstants.Success)
        set(handles.btn_OpnReport, 'Enable', 'on');
        set(handles.box_Sim, 'BackgroundColor', 'green');
        set(handles.box_Sim, 'String', 'OK');
    else
        set(handles.btn_OpnReport, 'Enable', 'off');
        set(handles.box_Sim, 'BackgroundColor', 'red');
        set(handles.box_Sim, 'String', '');
    end

     guidata(hObject, handles);
end

% falseInfoBoxes sets InfoBoxes red and no string
% called if no project is selected ...
% or Dropdown-Selection is not same as txt_WorkPath
function falseInfoBoxes(hObject, handles)
    set(handles.box_Proj, 'BackgroundColor', 'red');
    set(handles.box_CtrMdl, 'BackgroundColor', 'red');
    set(handles.box_TsNet, 'BackgroundColor', 'red');
    set(handles.box_Sim, 'BackgroundColor', 'red');
    set(handles.box_Proj, 'String', '');
    set(handles.box_CtrMdl, 'String', '');
    set(handles.box_TsNet, 'String', '');
    set(handles.box_Sim, 'String', '');
    guidata(hObject, handles);
end

% selItem gets the selected item of the parameter dropdown
% returns '' if selection is defVal
function selItem = getDrpSelItem(dropdown, defVal)
```

```
    selItem = '';
    items = get(dropdown,'String');
    drp_index = get(dropdown,'Value');
    drp_item = items{drp_index};
    if not(strcmp(drp_item, defVal))
        selItem = drp_item;
    end
end

%disableButtons, when no project is selected
function disableButtons(hObject, handles)
    set(handles.btn_RUN, 'Enable', 'off');
    set(handles.btn_ModiTsNet, 'Enable', 'off');
    set(handles.btn_ModiCtrMdl, 'Enable', 'off');
    set(handles.btn_SelCtrMdl, 'Enable', 'off');
    set(handles.btn_SelAppl, 'Enable', 'off');
    set(handles.btn_SelTsNet, 'Enable', 'off');
    set(handles.btn_DelProj, 'Enable', 'off');
    set(handles.btn_SavProjAs, 'Enable', 'off');
    set(handles.btn_OpnProj, 'Enable', 'off');
    set(handles.drp_SelSheet, 'Enable', 'off');
    guidata(hObject, handles);
end

% isXlsTsNet, returns 0 if the Excel-File is TsNet, 1 if is other Excel
function TsNetState = isXlsTsNet(hObject, handles)
    TsNetState = 0;
    if not(isempty(get(handles.txt_TsNetPath, 'String')))
        try
            [num, txt] = xlsread(get(handles.txt_TsNetPath, 'String'), ...
            GuiConstants.TsnSheet, GuiConstants.TsnCell);
            if  not(strcmp(txt, GuiConstants.TsnVal))
                TsNetState = 1; % cell is not as expected
            end
        catch
           TsNetState = 1;
        end
    end
end
```

11.2.2 IMSES_GUI.fig (Ansicht mit GUIDE)



11-2: Ansicht in GUIDE

11.2.3 checkErr.m (vollständig neu)

```
function checkErr(ErrState, handles, hwaitbar)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   (C) Copyright by Siemens Schweiz AG, Building Technologies Group,
%       HVAC Products, 2015
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Project                     : IMSES
%   Target Hardware             : PC
%   Target Operating System     : WinXP / Win7 Console
%   Language/Compiler           : Matlab 2010 and higher
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Workfile                    : checkErr.m
%   Author                      : Dominik Zgraggen
%   Version                     : v1.0
%   Date                        : 10-April-2015
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Matlab Informations
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:
%   checkErr evaluate the State of the Simulation process, if there is
%   an Error, it aborts the Simulation and gives a Message to GUI.
```

```
%   This function gets called by TSNet_Test.m after every Step
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function/Interface:
%%  - ErrState  is the Value of the Simulation Error handling (see also
%     GuiConstants)
%  - handles  structure with handles and user data (see GUIDATA)
%  - hwaitbar  Object to show the Simulation process to the User
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Revision History
%   (Put meaningful comments in SourceSafe for log below!)
%   (Please remove blank lines and very old comments!)
%
%   Document Creation (For IPA)
%   2015-04-10  Dominik Zgraggen, 5559
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    switch ErrState
        case {GuiConstants.NoError}
            set(handles.txt_State, 'String', GuiConstants.ErrOK);

        case {GuiConstants.TsNet_ReadErr} % in TSNet_Import() -> xlsread
            closeSim(handles, hwaitbar, GuiConstants.ErrRead);

        case {GuiConstants.TsNet_Keywords} % in TSNet_Import() -> getInd
            closeSim(handles, hwaitbar, GuiConstants.ErrKeyW);

        case {GuiConstants.CtrMdl_LoadErr} % TSNet_Sim() -> load_system
            closeSim(handles, hwaitbar, GuiConstants.ErrLoadCtrMdl);

        case  {GuiConstants.SimRunErr}  % in TSNet_Sim() -> sim
            closeSim(handles, hwaitbar, GuiConstants.ErrSim);

        case {GuiConstants.Report_WriteErr} % in TSNet_Report() -> xlswrite
            closeSim(handles, hwaitbar, GuiConstants.ErrRep);

        otherwise
            closeSim(handles, hwaitbar, GuiConstants.ErrUnkown)
    end
end

% closeSim gets called if simulation is going to be aborted
% these things should be done, before returning to GUI
function closeSim(handles, hwaitbar, State)
        set(handles.box_Sim, 'BackgroundColor', 'red');
        set(handles.box_Sim, 'String', '');
        set(handles.txt_State, 'String', State);
        delete(hwaitbar)
        error(State);
end
```

11.2.4 TSNet_Test.m (abgeändert Fremdcode)

```
function TSNet_Test(XLSName,SheetName,MdlName,handles)
……
%% Initiate TestFolder
try
    if isdir([handles.GuiProperties.ProjectPath '\' GuiConstants.TsNetDir])
        TSNetTestStruct.TestFolder = [handles.GuiProperties.ProjectPath '\'
GuiConstants.TsNetDir];        % path exists
    end
catch
        TSNetTestStruct.TestFolder=cd;
end
addpath(TSNetTestStruct.TestFolder);

%% Create a waitbar object to report the progress
hwaitbar = waitbar(0,'Import TsNet-File');

%% Error State init
ErrState = 0;

%% Import Data
[TSNetTestStruct, ErrState]=TSNet_Import(TSNetTestStruct);
checkErr(ErrState, handles, hwaitbar)

%% Run Simulation
% update waitbar
waitbar(1/4,hwaitbar,'Running TsNet Test..');

% run simulation
[TSNetTestStruct, ErrState]=TSNet_Sim(TSNetTestStruct);
checkErr(ErrState, handles, hwaitbar)

%% Evaluation of Results
 % update waitbar
waitbar(2/4,hwaitbar,'Evaluate Results...')
[TSNetTestStruct]=TSNet_Evaluation(TSNetTestStruct);

% update waitbar
waitbar(3/4,hwaitbar,'Creating Report..');
[ErrState] = TSNet_Report(TSNetTestStruct);
checkErr(ErrState, handles, hwaitbar)

% update waitbar
waitbar(4/4,hwaitbar,'Report created, TSNet test finished..');

if ErrState == 0
    set(handles.txt_State, 'String', GuiConstants.Success);
end
% Delete the waitbar object
delete(hwaitbar);

end % FUNCTION
```

11.2.5 TSNet_Import.m (abgeändert Fremdcode)

```
function [TSNetTestStruct, ErrState] = TSNet_Import(TSNetTestStruct)
……
try
    % Read the testcase sheet to txt (only txt data) and raw (everything)
    % for changing cell names
    [~, txt, raw]=xlsread(TSNetTestStruct.XLSName,TSNetTestStruct.SheetName);
    ErrState = 0;
catch
    ErrState = GuiConstants.TsNet_ReadErr;
    return
end

 %% Find coordinates of keywords
try
        % template version 0.7, adjust this for newer template versions

        [indXWT,indYWT]=getInd(txt,{['Waitingtime' char(10) 'after
Inputs'];});
        [indXIn,~]=getInd(txt,{'In';});
        [indXOut,~]=getInd(txt,{'Out';});
        [indXEnd,~]=getInd(txt,{'End';});
        [~,indYPrio]=getInd(txt,{'Priority (Number / All / empty)';});
        [~,indYPropID]=getInd(txt,{'Property-ID';});
        [~,indYObjID]=getInd(txt,{'Object-ID';});
        [~,indYObjType]=getInd(txt,{'Objecttype';});
        [indXStep, indYStep]=getInd(txt, {['Action', char(10),'*Comment is
mandatory*'];});
        indYTSD=3; % Test step description
        ErrState = 0;
catch
   ErrState = GuiConstants.TsNet_Keywords;
   return
end
……
```

11.2.6 TSNet_Sim.m (abgeändert Fremdcode)

```
function [TSNetTestStruct, ErrState]=TSNet_Sim(TSNetTestStruct)
……

%% Load System
try
    load_system(TSNetTestStruct.MdlName);
    ErrState = 0;
catch
   ErrState = GuiConstants.CtrMdl_LoadErr;
   return
end
……
```

11.2.7 TSNet_Report.m (abgeändert Fremdcode)

.....

```
try
    %Write to XLS of test
    xlswrite(XLSName,TSNetTestStruct.XLSOut,xlsSheetName);
    ErrState = 0;
catch
    ErrState = GuiConstants.Report_WriteErr;
    return
end

end
```

11.2.8 GuiConstants.m (vollständig erstellt)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   (C) Copyright by Siemens Schweiz AG, Building Technologies Group,
%       HVAC Products, 2015
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Project                     : IMSES
%   Target Hardware             : PC
%   Target Operating System     : WinXP / Win7 Console
%   Language/Compiler           : Matlab 2010 and higher
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Workfile                    : GuiConstants.m
%   Author                      : Dominik Zgraggen
%   Version                     : v1.0
%   Date                        : 10-April-2015
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Matlab Informations
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Description:
%   This Class contains all Constants needed for IMSES_GUI.m and checkErr.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function/Interface:
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Revision History
%   (Put meaningful comments in SourceSafe for log below!)
%   (Please remove blank lines and very old comments!)
%
%   Document Creation (For IPA)
%   2015-04-10  Dominik Zgraggen, 5559
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 classdef GuiConstants
    properties (Constant)

        % is Excel TsNet?
        TsnCell = 'A1';
        TsnVal = 'Application Test';
        TsnSheet = 'Config';

        % Foldernames of Projects
        ApplDir = 'Application';
        CtrMdlDir = 'Control_Model';
```

```
TsNetDir = 'TsNet';
AddDir = 'Additional';

% Simulation Error handling
NoError = 0;
TsNet_ReadErr = 1;        % TsNet_Import
TsNet_Keywords = 2;       % TsNet_Import
CtrMdl_LoadErr = 3;       % TsNet_Sim
SimRunErr = 4;            % TsNet_Sim
Report_WriteErr = 5;      % TsNet_Report

% Gui Texts
DefProjSel = 'Select existing Test-Project...';
DefSheetSel = 'Select the Excel-Sheet...';
CtrMdlSel = ['Bitte Modell auswählen, welches verwendet'...
    'werden soll (*.mdl)'];
TsNetSel = 'Bitte TsNet-Excel-Sheet auswählen';

%checkErr.m Strings
ErrOK = '...';
ErrRead = 'TsNet: Read Error!';
ErrKeyW = 'Could not find Keywords in TsNet!';
ErrLoadCtrMdl = 'Could not Load Control-Model';
ErrSim = 'Simulation failed!';
ErrRep = 'Could not write to Report-File';
ErrUnknown = 'unknown Error!';
Success = 'Simulation Successfully!';

% Message-Box Strings
ProjStrucTitle = 'invalid Project';
ProjStruc = ['This Selection is not a valid Project. It has' ...
     ' not got the Project-Folder-Structure'];
NoMdlTitle = 'Wrong Filetype: need to be *.mdl';
NoMdl = 'The selected File needs to be a Simulink-Model-File';
NoXlsTitle = 'Wrong Filetype: need to be *.xls or *.xlsx';
NoXls = 'The selected File needs to be a Excel-File';
ErrOpnFile = 'Could not open File: ';
ErrOpnFileTitle = 'Error open File';
OpnProj = 'Could not open Path: '
OpnProjTitle = 'Error Open Folder ';
FailRun = 'Input is not as expected: ';
FailRunTitle = 'Not ready for Simulation';
FileNotExist = 'A File does not exist anymore';
XlsNoTsn = 'This Excel-File has no valid TsNet-Structure';

% Help-Button-functions
% HELP-Button
HelpMes = ['Firstly, you should define where you '...
    'Work. A Test-Project collects your Files for a Test. ' ...
    char(10) 'After that you can select the Control-Model and'...
    ' the TsNet-File with the Sheet that contains the Testcase.'...
    ' When RUN is pressed, the simulation will be made and' ...
    ' after that you can evaluate the Report-File.'];
HelpMesTitle = 'General Help';

% [?] Projects
ProMesTitle = 'Help Test-Project-Selection';
ProMes1 = ['Drop-Down - Select existing Test-Project...:' ...
    char(10) 'In the Drop-Down you can see all Test-Projects,' ...
    ' which are available. If you select one and then click' ...
```

```
    ' "Load", the Test-Project and its Files will be loaded. ' ...
    char(10) char(10)];
ProMes2 = ['Name and Path of Projects:' ...
    char(10) 'All Test-Projects are located in the'...
    '"Projects"-Folder, which should be where your IMSES_GUI.m'...
    ' is. (Name of Test-Project = Name of Subfolder in'...
    ' "/Projects")' char(10) char(10)];
ProMes3 = ['Project-Subfolders:' ...
    char(10) 'All Test-Projects are located in the'...
    'Every valid Project contains the four Subfolders'...
    ' "Additional", "Application", "Control_Model" and "TsNet"'...
    char(10) char(10)];
ProMes4 = ['Create Project:' char(10) 'Type in a Name, then'...
    ' the folders will be created and your Work-Path should'...
    ' be set' char(10) char(10)];
ProMes5 = ['Save Project As:' char(10) 'If you have opened a'...
    ' existing Project, you can save it with a different name.'...
    ' So all files of the older Project will be copied to the '...
    ' newer one.'  char(10) char(10)];

% [?] Control-Model
MdlMesTitle = 'Help Control-Model Selection or Generation';
MdlMes1 = ['Control-Model:' char(10) ...
    'The Control-Model is a simulink-model containing two'...
    ' XFB-Blocks called AB_ComInterf and BA_ComInterf_Out and' ...
    ' the application-model-reference. If the Control-Model '...
    ' does not contain those XFB-Blocks and a referenced ' ...
    ' application-model, the test will fail. If you have done' ...
    ' the application-import and control-model-generation with' ...
    ' this tool, this should be in place.' char(10) char(10)];
MdlMes2 = ['Select Control-Model:'  ...
    char(10) 'If Control-Model-Path is empty, use the Button - '...
    ' Select existing Control-Model' ...
    ' If you select an existing Control-Model, which is not' ...
    ' located in your work-path, it will be copied into the' ...
    ' specific folder.' char(10) char(10)];
MdlMes3 = ['Select Application for Import:' char(10) ...
    'The Application is a ABT-Export as ZIP.' char(10) char(10)];
MdlMes4 = ['Import & Generate (no existing Control-Model):' ...
    char(10) 'If you have selected a ZIP, this will be' ...
    ' imported. The Control-Model will then be automatically' ...
    ' be generated with this import. See Logfile if the Import' ...
    ' fails.' char(10) char(10)];

% [?] TsNet
TsnMesTitle = 'Help TsNet-File and Excel-Sheet';
TsnMes1 = ['Select TsNet-File:' ...
    char(10) 'If TsNet-File-Path is empty, use the Button' ...
    ' - Select TsNet-File. If you select a TsNet-File, which' ...
    ' is not located in your Work-Path, it will be copied into' ...
    ' the specific Folder.'  char(10) char(10)];
TsnMes2 = ['Drop-Down - Select the Excel-Sheet...'  ...
    char(10) 'In the Drop-Down Select the Excel-Sheet you' ...
    ' must define, which Table / Sheet contains the Testcase' ...
    ' to be tested. First you need to select the File with' ...
    ' the Button Select TsNet-File' char(10) char(10)];
TsnMes3 = ['Modify TsNet-File:' char(10) ...
    ' Open the the Excel-File with this Button to look at' ...
    ' Sheets and also make changes.' char(10) char(10)];
```

```
        % [?] RUN - Simulation
        SimMesTitle = 'Help Simulation and Evaluation';
        SimMes1 = ['RUN button is not enabled:' ...
            char(10) ' - Control-Model needs to be selected' char(10)...
            ' - TsNet-File selected' char(10) ...
            ' - Test-Sheet in Drop-Down selected' char(10) char(10)];
        SimMes2 = ['Could not Load Control-Model' char(10) ...
            'Try the following things:' char(10) ...
            ' - open the Control-Model in Simulink check the' ...
            ' referenced Application-Model or follow the' ...
            ' error-message if there is one' char(10) ...
            ' - close GUI, use matlab-command - clear all, then' ...
            ' reopen GUI.' char(10) ...
            ' - or rename the Control-Model (This error can be' ...
            ' caused by a Model, which has same name and was' ...
            ' opened before.)' char(10) char(10)];
        SimMes3 = ['Report:' char(10) 'The Report is a Excel-Sheet,'...
            ' which has been added to your selected Excel-File. It' ...
            ' has the prefix R_, open it with - Show Report' ...
            char(10) char(10)];
    end
end
```