Bachelorarbeit

Modellvergleich mit Simio

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühlingssemester 2018

Autoren: Philipp Bütikofer
Anthony Delay

Verantwortlicher: Prof. Dr. Andreas Rinkel
Betreuer: Lukas Kretschmar

1. Abstract

Diese Bachelorarbeit beschäftigt sich mit dem Vergleich von Simulationsmodellen. Die Problemstellung ergibt sich aus den Defiziten der Simulationssoftware Simio. Momentan ist es in Simio nicht möglich, aus einem bestehenden Modellen verschiedene Varianten zu erzeugen und diese anschliessend miteinander zu vergleichen. Um Simio um diese Funktionalität zu erweitern, ist im Rahmen dieser Bachelorarbeit ein entsprechendes Add-In entwickelt worden.

Im Zentrum steht das entwickelte Add-In «ModelAnalyzer», welches es ermöglicht Modellvarianten zu erzeugen und anschliessend miteinander zu Vergleichen. Das Resultat eines solchen Modellvergleichs zeigt an, ob zwischen den verglichenen Modellen eine signifikante Abweichung besteht. Durch das Aufdecken signifikanter Abweichungen können anschliessend statistische Aussagen über die einzelnen Modellvarianten getroffen werden.

Während der Implementierung kam die Frage auf, in welcher Simulationsphase der ModelAnalyzer einzuordnen ist. Schnell wurde ersichtlich, dass diese Einordnung eine Weiterentwicklung der Simulationsphasen erfordert. Hierzu wurde ein spiralförmiges Modell entwickelt. Diese Spirale wiederholt sich bis das Simulationsmodell und die Realität übereinstimmen. Unterstützt wird dieser Vorgang durch den ModelAnalyzer, welcher Kernfunktionalitäten und Teilschritte der Simulationsspirale übernimmt.

Als Resultat dieser Bachelorarbeit wurde ein neuer Ansatz des Simulationsvorgehens geschaffen. Unterstützt wird dieses Modell durch das Simio-Add-In «ModelAnalyzer», welches eine Entscheidungsbasis für den Hypothesentest schafft.

Inhalt

2. Management Summary

Das Ziel dieser Bachelorarbeit ist die Implementierung eines Add-Ins für die Simulationssoftware Simio. Die Kernfunktionalität dieses Add-Ins liegt im Vergleich von Simulationsmodellen. Dadurch sollen signifikante Abweichungen zwischen mehreren Modellen aufgezeigt werden können. Diese Bachelorarbeit gliedert sich in zwei Teile. Der erste Teil befasst sich mit dem Modellvergleich und dem dazugehörigen Modellierungsprozess. Innerhalb des zweiten Teil wird auf die Implementation des Add-Ins «ModelAnalyzer» eingegangen, welcher Teile dieses Modellierungsprozesses abbildet. Zum Abschluss werden die Funktionsweise des ModelAnalyzers und der Simulationsspirale an einem Beispiel aufgezeigt.

2.1 Modellvergleich

Zu Beginn wird auf die Grundbegriffe der Simulation und Phasen eines Simulationsprojektes eingegangen. Anschliessend werden Defizite in Simio analysiert, aus welchen sich die Anforderungen des entwickelten Add-Ins ergeben haben. Momentan ist es in Simio nur durch aufwändige Workarounds möglich, verschiedene Varianten von Modellen anzulegen. Die Möglichkeit eines Modellvergleichs ist bisher inexistent.

Durch diese Anforderungen und Defizite stellt sich die Frage, an welcher Stelle sich die Erzeugung der Modellvarianten und deren Vergleich in die bestehenden Simulationsphasen eingliedern lassen. Die Antwort auf diese Frage liegt in der Weiterentwicklung der bis anhin genutzten Simulationsphasen. Diese Weiterentwicklung hat sich über mehrere Stufen hinweggezogen.

Das Resultat stellt ein spiralförmiges Modell dar, welches über verschiedene Ebenen verfügt. Jede Ebene bildet dabei eine Modellvariante ab. Sämtliche dieser Varianten/Ebenen durchlaufen alle drei Simulationsphasen. Am Ende eines solchen Durchlaufs wird die Modellvariante validiert. Das Ergebnis dieser Validierung gibt das weitere Vorgehen vor. Entweder bildet das Modell die Realität im gewünschten Grad ab, was das Ende des Modellierungsprozesses heisst, oder eine Weiterentwicklung der Modellvariante ist nötig. Der weite Fall resultiert in einer weiteren Ebene. Als Grundlage für diese neue Variante/Ebene gilt der finale Zustand des Modells der darüberliegenden Ebene. Dieses Vorgehen läuft so lange, bis sich das Modell genügend an die Realität angenähert hat.

Weiter wird auf die mathematische Grundlage, ein Differenztest mit unabhängigen Stichproben, eingegangen. Um die Funktionsweise des Add-Ins und den konkreten Einsatz in diesem neuentwickelten Modellierungsprozess zu veranschaulichen, wurde das Vorgehen anhand eines einfachen Beispielprojekts durchlaufen.

2.2 Implementation «ModelAnalyzer»

Der zweite Teil umfasst das SAD (Software Architektur Dokument). Darin sind implementationsspezifische Aspekte und Entscheidungen festgehalten.

Kern dieser Arbeit bildet der ModelAnalyzer. Dieses Simio-Add-In bildet Teilschritte des im ersten Teil vorgestellten Modellierungsprozesses ab. Die Kernfunktionalität ist in zwei logische Komponenten aufgeteilt: Modellmanagement und Modellvergleich. Die Komponente «Modellmanagement» ermöglicht es, bestehende Simio-Modelle zu klonen sowie zu versionisieren. Durch das anschliessend Modifizieren und Weiterentwickeln eines solchen Klons entstehen die im Modellierungsprozess erwähnten Modellvarianten. Die zweite Komponente, Modellvergleich, setzt den Differenztest um. Die Responses der Modellvarianten werden gegen ein vorher definiertes Basis-Modell verglichen. Die Ausgabe dieses Vergleich zeigt signifikante Abweichungen des Basis-Modells im Kontrast mit den Modellvarianten auf. Der ModelAnalyzer bietet weitere Features, welche diesen Vorgang unterstützen wie beispielsweise die Versionierung und das Kommentieren der einzelnen Modellvarianten. Die Vergleichsergebnisse können einfach gefiltert und strukturiert werden. Weiter ist es nun möglich, modellübergreifende

Boxplots der einzelnen Responses zu erzeugen. Die Weiterverarbeitung der Vergleichsergebnisse ist durch eine Exportfunktion ermöglicht.

Im Zuge dieser Bachelorarbeit sind verschiedene weitere Implementationen entwickelt worden. Der Fokus lag dabei auf der Entkopplung der SimioAPI und dem ModelAnalyzer. Die Lösung ist als API-Adapter realisiert. Dieser kapselt die relevanten, benötigten Daten der SimioAPI. Durch die Architektur als Service, kann dieser leicht erweitert und von anderen Add-Ins genutzt werden. Zur Vereinfachung der Installation ist ein Installer entwickelt worden. Dieser führt den Benutzer durch eine graphische Oberfläche.

2.3 Schlussfolgerung und Ausblick

Die an den ModelAnalyzer gestellten Anforderungen sind abgedeckt. Durch dieses Add-In können nun Modelle, ohne Verluste oder Einschränkungen, geklont und versioniert werden. Durch den implementierten Modellvergleich können einfach signifikante Abweichungen ermittelt werden. Durch den effektiven Einsatz des ModelAnalyzers kann allfälliges Erweiterungspotential aufgedeckt werden.

Die entwickelte Simulationsspirale zeigt einen neuen Ansatz des Hypothesentests auf. Unterstützt wird diese Vorgehensweise durch den ModelAnalyzer. Durch das Einsetzen der Simulationsspirale kann ihre Tauglichkeit und Optimierungspotential ermittelt werden.

Der Entwicklung von Simio-Add-Ins an der HSR wurde durch die Implementierung des API-Adapters und des Installers eine neue Basis geschaffen. Dadurch, dass diese Plattformen leicht erweiterbar und zukunftsorientiert implementiert sind, wurden erste Hürden im Zusammenhang mit Simio und dessen API bereits überwunden.

3. Modellvergleich

3.1 Inhalt

3.2 Einleitung

Dieser Abschnitt zeigt die Grundbegriffe der Simulation und die verschiedenen Phasen eines Simulationsprojektes auf. Darauf aufbauend ist im Anschluss der Modellierungsprozess präzise beschrieben. Abschliessend ist der Modellvergleich aus mathematischer Sicht beschrieben.

3.3 Grundbegriffe der Simulation

Der Begriff der Simulation [1] ist weitläufig und schwer zu fassen. Grundsätzlich kann Simulation als Methode verstanden werden, welche es erlaubt, reale Systeme zu imitieren bzw. nachzubilden. Simulation kann in verschiedene Arten unterteilt werden: Monte-Carlo-Simulation, statische Simulation und die dynamische Simulation. Dieses Projekt befasst sich mit der diskreten Ereignissimulation (DES), welche in der Art der dynamischen Simulation einzuordnen ist. Die DES benutzt die Zeit um bestimmte Ereignisse hervorzurufen, welche wiederrum den Systemzustand beeinflussen/bestimmen. Durch DES wird ein offenes System in ein geschlossenes überführt. Dies Aspekte der Realität werden jedoch mit eingebracht. Da bei DES der Zufall bzw. die Wahrscheinlichkeit in ein Modell miteingebracht wird, kann nach genügender Anzahl Durchläufe eine Aussage über zu erwartende Systemzustände gemacht werden. Durch diese Stärke ist der Anwendungsbereich heutzutage auch entsprechend gross:

- Prozesse der Logistik
- Abläufe in der Produktion
- Prozesse mit grossem Ressourcen-Aufkommen

Eine Simulation kapselt ein bestimmtes System, welches genau definierte Grenzen und Schnittstellen zur Umwelt aufweist und somit in einen Systemkontext eingebettet ist. Innerhalb des Systems können verschiedene Komponenten eingesetzt werden, welche terminal, also abgeschlossen, sind oder wiederrum Subsysteme abbilden. Ein System wird an seiner Komplexität und Kompliziertheit gemessen. Die einzelnen Systemkomponenten dienen als Hüllen für Prozesse. Jeder Prozess besteht aus einer Abfolge von verschiedenen Aktivitäten und entsprechenden In- und Outputs.

Durch Simulation können Modelle erzeugt werden, welche die reale Welt abstrahieren. Präziser ist ein Modell eine Menge von Annahmen und Annäherungen, welche eine Aussage über das Systemverhalten machen. Eine Simulation stellt sich als billiger und effizienter heraus, anstelle des realen Systems das Modell zu analysieren. Durch den, mit Simulation in Verbindung stehenden, Modellierungsprozess wird bereits ein grosses Verständnis über das Systemverhalten erworben.

Um die Simulation auszuwerten und zu untersuchen, werden Experimente geplant und durchgeführt. Ein Experiment stellt einen Test oder eine Untersuchung dar, durch welches eine Hypothese überprüft wird. Dies geschieht durch das Ausführen des Experiments unter kontrollierten Bedingungen. Als Resultat kann bestimmtes Verhalten aufgedeckt werden. Als weiteren Schritt kann anschliessend, falls nötig, optimiert werden.

3.3.1 Gründe für Simulationsprojekte

Ein Simulationsprojekt wird gestartet, um eine Frage zu beantworten. Meist beinhaltet diese den Aspekt wie etwas verbessert/optimiert werden kann. Zu Beginn steht man vor drei Varianten um diese Frage zu beantworten:

- Mathematisches Modell
- Echte Experimente in der Realität
- Computersimulation

Häufig reichen jedoch mathematische Modelle oder Experiment in der Realität nicht aus, um diese Fragen zu beantworten. Die Simulation ermöglicht es jedoch, Entscheidungen aufgrund eines Modells

zu treffen. Somit kann durch Simulation eine Entscheidungsbasis geschaffen werden. Durch das Simulieren des Modells werden die anfangs gestellten Fragen nach und nach beantwortet.

Simulationsprojekte bringen zusätzlich folgenden Mehrwert mit sich:

- Analyse des zeitlichen Verhaltens
- Animation
- Prozesskomplexität/ Studium hochvariabler Prozesse
- Verständnis von Prozesswechselwirkungen eines Systems schaffen
- Kostengründe/ Das reale System existiert noch nicht

3.4 Validierung und Verifikation

Die Schritte der Validierung und der Verifikation [2] stellen zwei der fundamentalen Tätigkeiten innerhalb der Simulation dar, da sie sich mit den beiden Komponenten Modell und Realität beschäftigen. Dies führt daher, dass in der Simulation gleich mehrere Disziplinen zusammenfallen:

- Grundlagen der Experimentplanung und – Durchführung
- Modellbildung und Spezifikation (Methoden- und Domänenwissen erforderlich)
- Implementierung des Modells (erfordert das Beherrschen der Simulationssoftware)
- Auswertung

Die Validierung und die Verifikation prüfen diese Punkte auf ihre Korrektheit. Die Verifikation untersucht die Frage, ob das Modell gegenüber dessen Spezifikation «richtig» gebaut wurde, daher finden die Schritte zur Verifikation eines Modells auf der Stufe der Implementation statt. Als Ansätze zur Verifikation können folgende Punkte gesehen werden:

- Testszenarien als fester Bestandteil der Entwicklung und Implementierung
- Testen der einzelnen Komponenten
- Testen von Randwerten und Systemgrenzen
- Je präziser (formaler) die Beschreibung/Spezifikation des Modells ist, desto leichter ist die Verifikation

Schlussendlich wird ein Test gegen die Spezifikation durchgeführt. Dabei ist die Verifikation nicht als einmaliger Arbeitsschritt zu sehen, sondern als wiederkehrender, iterativer Vorgang. Falls die Verifikation negativ ausfällt, ist es wichtig, nicht das Modell, sondern die Spezifikation zu verändern. Die Schwierigkeit der Verifikation liegen in Aspekten wie z.B. Nebenläufigkeit.

Die Validierung beschäftigt sich mit der Frage, ob das System hinsichtlich des zu untersuchenden Fragebereichs richtig abgebildet wurde. Einfacher gesagt, bildet das Modell die Realität ab, ist das Verhalten des Modells so wie das des Originals. Validiert werden müssen die Modellogik und die Modellparameter (Inputdaten). Dabei ist die gesamte Architektur inklusive des Datenmaterials zu berücksichtigen. Auch die Validierung ist als iterativer Prozess zu verstehen. Dieser wird solange durchgeführt, bis eine hinreichende Genauigkeit erreicht wird.

3.5 Simulationsphasen

Ein Simulationsprojekt wird in drei Phasen [1] unterteilt: Process Analysis, Modelling und Programming. Die erste Phase beinhaltet die Analyse des realen oder imaginären Systems. Diese Unterscheidung zwischen den Systemen kann folgendermassen beschrieben werden: ein reales System existiert bereits in einer physischen Form, wohingegen ein imaginäres erst in der Planung existiert. Durch die Analyse des Systems können so verschiedene Prozesse und Subsysteme isoliert werden. Die Erhebung von empirischen oder hypothetischen Daten fällt ebenfalls in diese erste Phase.

Das Resultat der anschliessenden zweiten Phase ist es, die gesammelten Daten und Anforderungen in ein formales Modell zu überführen. Typischerweise werden dazu formale und mathematische Methoden genutzt. Die Das Gesamtsystem wird in verschiedene Subprozesse zerlegt. Ein formales Modell wird beispielsweise in BPMN beschrieben.

In der dritten Phase wird das erarbeitete, formale Modell nun innerhalb einer Simulationssoftware programmiert/umgesetzt. Der Simulator bietet die Simulationslaufzeitumgebung für das nun programmierte Modell. Durch das Simulieren des Modells können so erste Resultate ermittelt werden. Dieses programmierte Modell ist nun durch Experimente zu ergänzen. Die Experimentresultate geben die gesuchten Antworten.

Die Simulationsphasen bauen iterativ aufeinander auf. Je nach Ausfall der Resultate kann nun iterativ das formale Modell weiterangepasst und simuliert werden. Folgende Grafik stellt die Simulationsphasen und deren Zusammenhänge dar.

Dieses Phasenmodell bildet die Grundlage zum Modellierungsprozess [3], welcher in den folgenden Kapiteln erörtert wird.

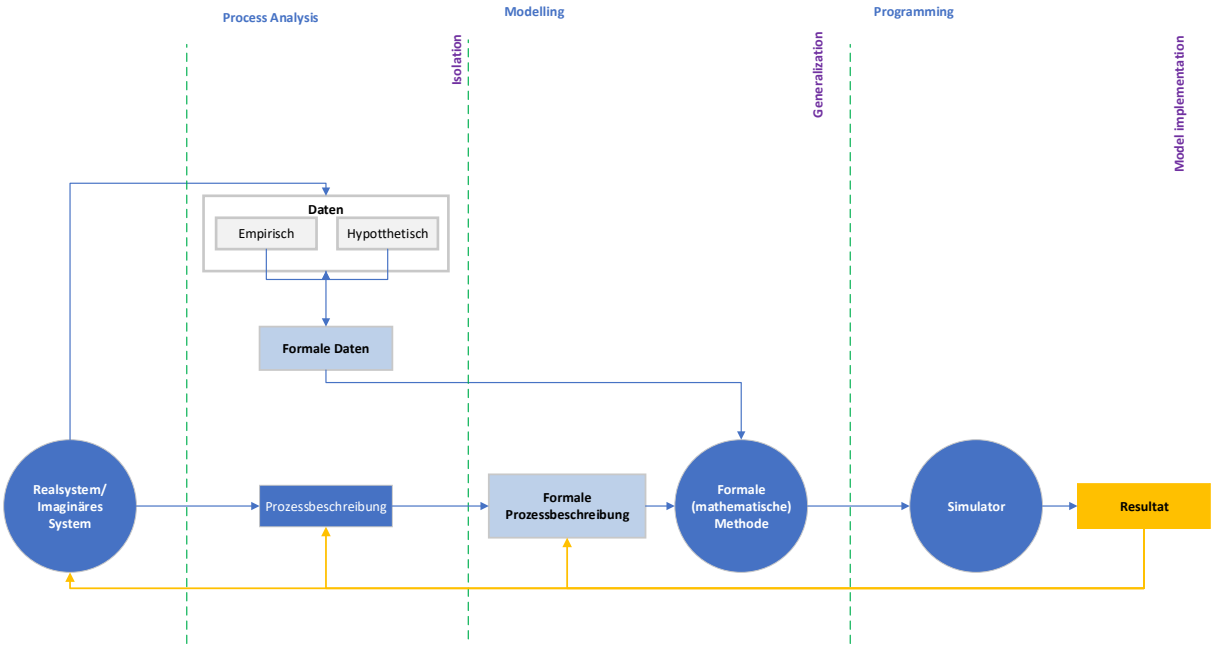


Abbildung 1: Simulationsphasen[1]

[1] Übernommen und neu gezeichnet aus dem Modul SMS [1]

3.6 Modellierungsprozess

Dieses Kapitel soll einerseits Defizite von Simio und deren Lösungsansätze aufzeigen, andererseits den Modellierungsprozess, welcher als Grundlage des entwickelten Addin dient, darstellen und erklären.

3.6.1 Defizite in Simio

Dem Benutzer ist es innerhalb von Simio nicht möglich, verschiedene Modellvarianten innerhalb derselben Projektstruktur, effizient zu erstellen. Es existieren verschiedene Workarounds, welche sich jedoch als mühselig und wenig gewinnbringend herausstellen. Weiter können, auf diese Weise erzeugte Modellvarianten nicht automatisiert miteinander verglichen werden.

Aus diesen Defiziten können einerseits ein theoretischer Modellierungsprozess, sowie klare Anforderungen an das Addin abgeleitet werden.

Das zu entwickelnde Addin muss in der Lage sein, bestehende Modell identisch zu klonen und diese in die bestehende Projektstruktur einzubinden. Der entstandene Klon darf keinen Einschränkungen unterliegen, d.h. alle Simio-Funktionalitäten müssen anwendbar sein. Jeder Klon muss über eine Versionsnummer verfügen.

Die zweite Komponente umfasst den Modellvergleich, welcher folgenden Anforderungen erfüllen soll. Der Vergleich erstreckt sich über verschiedene Modell hinweg. Verglichen werden dabei Response-Resultate. Durch das Festlegen eines Basismodells bzw. Basisszenario werden alle verbleibenden Modelle gegen diese Basis verglichen. Nachdem der Vergleich abgeschlossen ist, kann eine Aussage über signifikante Abweichungen zwischen dem Basismodell und den Klonen getroffen werden.

3.6.2 Abstraktion des Modellierungsprozesses

Während der Entwicklung hat sich der Modellierungsprozess stetig weiterentwickelt und verschiedene Varianten und Stufen durchlebt. Die einzelnen Ansätze werden an dieser Stelle in chronologischer Reihenfolge kurz dargelegt, eine detailliertere Beschreibung des finalen Modellierungsprozesses ist am Ende zu finden.

3.6.3 Erste Entwicklungsstufe: Subprozess

Zu Beginn wurde die gesuchte Funktionalität lediglich als Subprozess gesehen. Dieser wurde dabei in die letzte Simulationsphase eingebettet. Die Idee lag darin, dass bestehende Elemente der Simulationsphase angesprochen werden und der erweiternde Subprozess in die bestehende Phase eingeflochten wird. Das Resultat dieser Entwicklungsstufe zeigt folgende Grafik. Gut erkennbar sind hier die Versuche durch ein- und austretende Ereignisse auf bestehende Elemente der Simulationsphasen zuzugreifen.

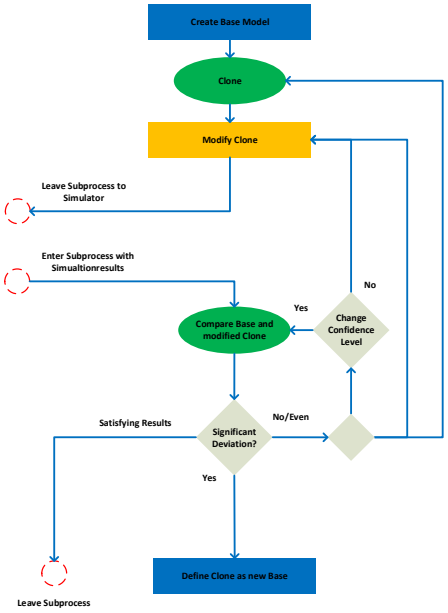


Abbildung 2: Modellierungsprozess als Subprozess

3.6.4 Zweite Entwicklungsstufe: Spirale

Während dieser Stufe wurde der Modellierungsprozess aus den Simulationsphasen herausgelöst und als abgeschlossenes System betrachtet. Dabei wurde der Fokus auf das iterative Klonen und Vergleichen eines Basismodells gelegt, was anschliessend in einer Art Spirale resultierte. Das Resultat ist in Abbildung 3 aufgezeigt.

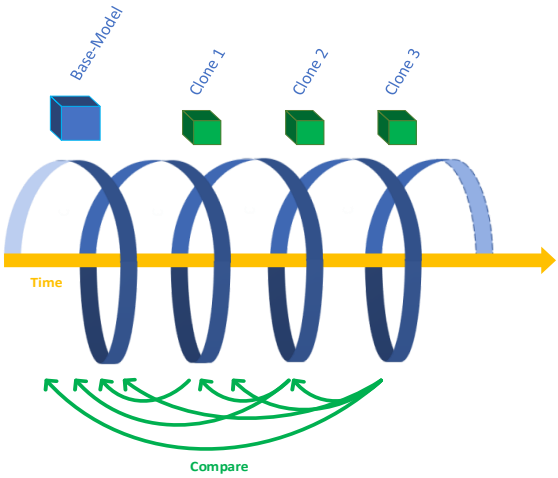


Abbildung 3: Spiralförmiger Modellierungsprozess

3.6.5 Dritte Entwicklungsstufe: Iteratives Bindeglied

Die dritte Stufe befasste sich wieder mit der Eingliederung des entwickelten, spiralförmigen Modellierungsprozesses. Als Basis dienten dabei die Resultate der ersten beiden Entwicklungsphasen. Hierbei wurde der Modellierungsprozess nicht als Subprozess innerhalb einer Simulationsphase gesehen, sondern als Bindeglied der rückkoppelnden Verbindungen zwischen den einzelnen Arbeitsschritten der Phasen. Die wesentlichste Erkenntnis dieser Phase liegt in der Tatsache, dass sich der Modellierungsprozess über alle Simulationsphasen hinwegzieht.

3.6.6 Finale Stufe: Aufspannung als Ebene

Aufgrund weiterführenden Überlegungen und den Erkenntnissen aus der dritten Phase, sind wir zum Schluss gekommen, dass der Modellierungsprozess nicht innerhalb der Simulationsphasen gesehen werden kann, sondern der Sachverhalt genau umgekehrt ist. Der Modellierungsprozess beinhaltet in jedem Schritt einen kompletten Durchlauf aller Simulationsphasen. Der detaillierte Aufbau des finalen wird im nächsten Abschnitt genauer dargelegt.

3.7 Simulationsspirale

Dieser Ansatz des Modellierungsprozesses basiert auf der Idee, den Prozess in eine weitere Dimension aufzuspannen. Der Grundgedanke besteht immer noch aus den beiden Bausteinen des Klonens und Vergleichens eines Modells, wird jedoch um die Verifikation und Validation erweitert. Die Grundidee, eine weitere Dimension in den Modellierungsprozess einzuführen und mit Ebenen zu arbeiten ist an andere mathematische Modelle angelehnt. In unserem Modell stellt der Nord- und Südpol, bzw. Start- und Endpunkt jeweils die Realität dar. Dies ist folgendermassen zu verstehen: Der Ausgangspunkt ist die Realität. Davon wird ein Modell gebildet, welches eine Anzahl von verschiedenen Varianten durchlaufen wird, was jeweils eine Ebene darstellt. Durch diese varianten wird der Detaillierungsgrad immer feiner und präziser. Dies bedeutet wiederrum, dass sich das Modell immer mehr dem Realsystem annähert, bis diese komplett abgebildet ist.

In der Simulationsspirale stellt jede Modellvariante eine eigene Ebene dar. Innerhalb dieser Ebene werden die Simulationsphasen wie in Kapitel 3.5 beschrieben vollständig durchlaufen. Als letzter Schritt einer solchen Iteration steht die Validation. Zu Beginn wird diese nicht in den meisten Fällen nicht besonders genau ausfallen. Daher wird eine neue Modellvariante erzeugt, welche wiederrum eine Ebene unter dem Original steht. Die Relation zwischen zwei aufeinanderfolgenden Ebene bilden die gewonnen Erkenntnisse, wobei die vorangehende Modellvariante die Basis für die Folgende darstellt. Diese fliessen anschliessend in die nächste Ebene und in die damit verbundenen Simulationsphasen ein.

3.7.1 Umsetzung

Zur übersichtlicheren Veranschaulichung werden die in Kapitel 3.5 vorgestellten Simulationsphasen auf die drei wesentlichen Bestandteile reduziert. Daraus ergibt sich folgender Ablauf.

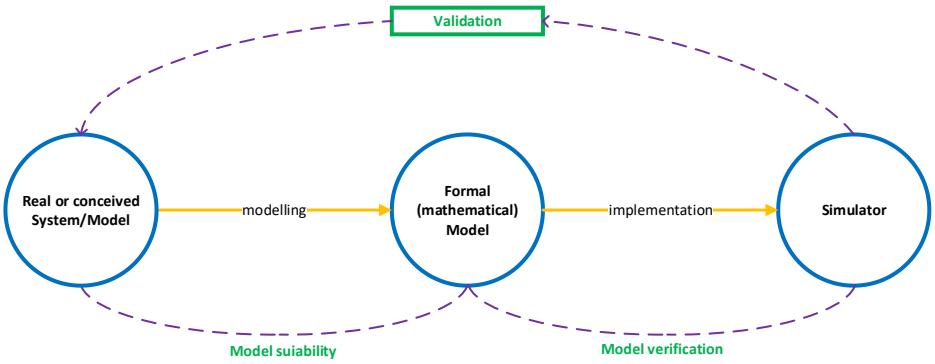


Abbildung 4: Reduzierte Simulationsphasen[2]

In der finalen Version des Modellierungsprozesses bildet der «normale» Simulationszyklus lediglich einen Teilschritt ab. Der dargestellte Prozess wird in einer Ebene aufgespannt. Eine solche Ebene beherbergt nun alle Simulationsphasen, welche entsprechend vollständig durchlaufen werden. Durch eine abschliessende Validierung und Auswertung wird dann eines von zwei Ereignissen ausgelöst, welche wiederrum verschiedenen Kriterien unterliegen.

Falls die Validation als gültig, d.h. das Modell bildet die Realität zum gewünschten Grad ab, ausfällt, wird der Prozess als beendet angesehen. Diese Entscheidung wird getroffen, falls die Resultate und die Verifikation ebenfalls den Zielanforderungen entsprechen. Das zweite Ereignis wird entweder durch eine negative Validation oder durch unzureichende Zufriedenheit der erreichten Resultate ausgelöst. Als weiterführender Schritt wird das momentane Modell geklont und anschliessend zu einer eigenständigen Modellvariante weiterentwickelt. Der in Abbildung 4 dargestellte Prozess wird daher um diese Entscheidung/Ereignisse erweitert.

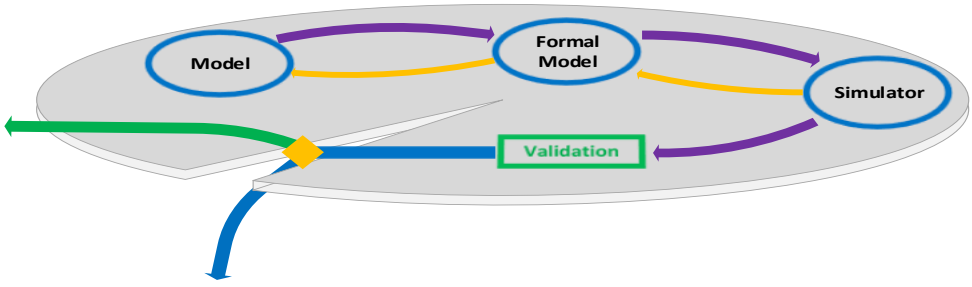


Abbildung 5: Einzelne Ebene der Simulationsspirale

Aus Sicht des Modellierungsprozesses wird eine weitere Ebene eingeführt. Diese behandelt nun innerhalb der drei Simulationsphasen den Klon der vorgehenden Ebene. Ein wesentlicher Bestandteil stellt

[2] Übernommen und neu gezeichnet aus dem Modul SMS [1]

das Bindeglied zwischen den Ebenen dar. Dieses bringt die gewonnenen Erkenntnisse aus der vorhergegangenen Ebene bzw. Modellvariante mit sich. Diese bilden wiederrum eine wichtige Basis. Denn als Ausgangslage einer beliebigen Ebene (ausschliesslich der der Ersten) bilden diese Erkenntnisse und nun den ersten Schritt der Simulationsphasen dar. Dies bedeutet also, dass auf dieser Ebene nicht von einem realen System ausgegangen wird, sondern vom Ergebnis der vorangehenden Ebene.

Dieser Ablauf kann sich nun mehrmals wiederholen, es entstehen immer weitere Ebenen. Dadurch, dass die Ebenen miteinander verbunden sind, werden sich die einzelnen Modellvarianten in ihrem Detailgrad immer mehr verfeinern. Im gleichen Zuge werden sich die Modellvarianten immer weiter an die Realität annähern. Diese Annäherung bzw. das Modell in den Zustand der Realität zu bringen, stellt die Zielsetzung des gesamten Prozesses dar. Abbildung 6 zeigt das Konzept des Modellierungsprozesses auf.

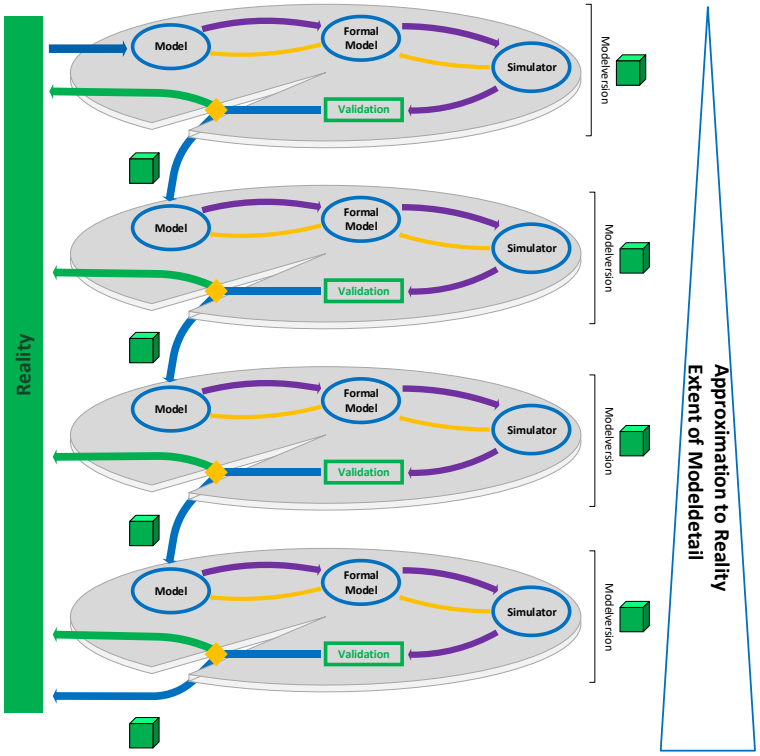


Abbildung 6: Simulationsspirale

3.8 Mathematische Grundlage des Modellvergleichs

Die mathematische Grundlage zum Modellvergleich bildet ein Differenztest mit unabhängigen Stichproben [4]. Dies ist ein statistischer Test, welcher eine Entscheidung ermöglichen soll. Ein generelles Vorgehen eines solchen Tests lautet wie folgt:

1. Hypothese aufstellen, welche bewiesen werden soll (Modelle sind gleich)
2. Nullhypothese aufstellen, welche wiederlegt werden soll, um die Hypothese nicht zu verwerfen
3. Festlegen des Konfidenzintervalls, definiert **Signifikanzzahl α**
4. Bestimmung der Testgrösse **X** (z.B. Stichprobenmittelwert)
5. Bestimmung der Schranken x_{krit} für
   a. **Einseitiger Test $P(X > x_{krit}) = \alpha$**
   b. **Zweiseitiger Test $P(|X| > x_{krit}) = \frac{\alpha}{2}$**
6. Werte für $\boldsymbol{F}(\frac{x_{krit} - \mu}{\sigma})$ aus Tabelle auslesen
7. Annahmebereich berechnen

$$c = \pm z \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$$

8. Nullhypothese annehmen, wenn:

$$-c \leq \bar{x} - \bar{y} \leq +c$$

Dadurch, dass es sich in diesem Fall um einen Differenztest mit unabhängigen Stichproben handelt, können die Anzahl der Replikationen oder Durchläufen entsprechend unterschiedlich sein.

3.8.1 Modellvergleich mit Simio

Der Vergleich zwischen Modellen liegt den Responseresultaten zugrunde. Wie aus dem Modellierungsprozess ersichtlich ist, werden keine Parameter oder spezifische Elemente innerhalb der Modelle verglichen, sondern die Resultate der verschiedenen Responses. Innerhalb eines Experimentes befinden sich sogenannte Scenarios. Ein Scenario definiert für ausgewählte Properties einen spezifischen Parameter. Durch das mehrmalige ausführen, gesteuert durch die Anzahl Replikationen, können so die Auswirkungen der einzelnen Parameter auf das System untersucht werden. Eine Response greift auf eine gültige *Expression* eines Elementes zu, typischerweise einen Key Performance Indikator, und kapselt anschliessend die Scenarioresultate. Über die Responseresultate können anschliessend statistische Aussagen über den Simulationsdurchgang gemacht werden.
Eine Response kapselt nicht nur einen einzigen Wert, sondern verschiedene Werte des gesamten Experimentdurchlaufs. Die Responseresultate umfassen folgende Werte:

- Mean
- Median
- Minimum
- Maximum

Aufgrund dieser Wertesammlung, kann leicht ein Boxplot erzeugt werden, welcher einen visuellen Aufschluss über eine Response gibt.

Der hier umgesetzte Modellvergleich greift auf die Responseresultate der einzelnen Scenarien zu. Dabei können diese Vergleiche einerseits innerhalb des gleichen Modells, aber auch modellübergreifend stattfinden. Zur Berechnung werden die Mittelwerte (Mean) der zu vergleichenden Responses gegenübergestellt.

3.8.2 Konfidenzintervall und Signifikanz

Auf der Basis des Modellvergleichs soll der Anwender anschliessend eine Entscheidung treffen können. Das Ziel des Vergleichs ist es, signifikante Abweichungen von einem Modell im Kontrast mit einem anderen aufzuzeigen.

Die hier gesuchte statistische Signifikanz hängt von der Stichprobengrösse ab. Falls nur eine kleine Stichprobengrösse zur Verfügung steht, kann diese unter Umständen nicht die Grundgesamtheit genügend repräsentieren. Eine Abweichung gilt dann als signifikant, falls die Stichproben so stark von der Nullhypothese abweichen, dass diese Annahme anschliessend verworfen wird. Im Kontext dieser Arbeit bedeutet dies, dass zwei Responseresultate sich so stark voneinander unterscheiden, dass die Abweichung als signifikant gilt.

4. Software Architecture Document

4.1 Inhalt

4.2 Einleitung

Dieser Teil gibt Aufschluss über die Implementierung des ModelAnalyzer, Adapter.SimioAPI und des Simio Tools Installer. Zu Beginn werden analytische Aspekte wie Anforderungen und Use-Cases definiert. Anschliessend wird auf die konkrete Implementierung eingegangen. Zum Abschluss werden UI-technische Entscheidungen und Prozesse aufgeführt.

4.3 Eingesetzte Technologien

Sämtliche Teilimplementierungen des ModelAnalyzers und Adapter.SimioAPI basieren auf dem .NET Framework 4.6.1.

	Bezeichnung	Version
Libraries und NuGet Packages	SimioAPI	10.165.15447
	Mathnet.Numerics	4.4.0
	SQLite	1.4.118
	QxyPlot	1.0.0
	WIX Toolset	3.1.1
	Unity	5.8.6

4.4 Funktionale Anforderungen

Das Add-In «ModelAnalyzer» soll folgende funktionale Anforderungen abdecken. Diese wurden zu Beginn des Projekts ermittelt und spezifiziert. Die kursiv und fett geschriebenen Anforderungen sind massgebend und stellen die Kernfunktionalität des Add-Ins dar und werden somit höher gewichtet.

Anforderung	Beschreibung
Modellvergleich ***Signifikante Abweichungen anzeigen***	Signifikante Abweichungen von zwei Modellen können durch einen Differenztest mit unabhängigen Stichproben errechnet werden.
Basis-Modell festlegen	Ein Modell muss als Basis festgelegt werden können. Dieses Basismodell gilt als Grundlage für die Berechnungen.
Szenariovergleich/Experimentvergleich (im gleichen Modell)	Vergleiche von Szenarien und Experimente innerhalb eines Modells müssen nach gleicher Berechnung unterstützt sein.
Klonen von Modellen	Bestehende Simio-Modelle müssen identisch kopiert bzw. geklont werden können. Die Funktionalität und das Verhalten des Klons darf sich nicht vom Original unterscheiden.
Versionisierung von Modellen	Einzelne Modelle müssen mit einer Versionsnummer versehen werden können. Das Schema dieser Versionierung muss frei wählbar sein.
Konfidenzlevel setzen	Der Konfidenzlevel zur Berechnung der signifikanten Abweichungen muss steuerbar sein.
Export der Vergleichsergebnisse	Die Vergleichsergebnisse müssen in einer Form exportiert werden können.
Modellübergreifende Visualisierung	Responses müssen modellübergreifend als Boxplot visualisiert werden können.

4.5 Nicht-funktionale Anforderungen

Dadurch, dass dieses Projekt verschiedene Endprodukte umfasst, können so auch nicht-funktionale Anforderungen (NFR) abgedeckt werden. Auf die Umsetzung und Abdeckung der einzelnen NFRs wird jeweils in dem Kapitel deren Umsetzung vertieft eingegangen.

NFRs	Produkt/Aspekt
Deployment Extensibility	Simio Tools Installer
Dependency on other Parties Extensibility	Adapter.SimioAPI
Usability Look and Feel	ModelAnalyzer

4.6 Constraints

Dem Add-In sind verschiedene Einschränkungen gesetzt, welche in diesem Abschnitt dargelegt werden.

Plattform
Das Add-In wird lediglich auf Windows Betriebssystemen verfügbar sein. Grund dafür ist, dass Simio selbst auf diese Plattformen[3] gebunden ist. Die minimal Anforderung ist Windows 7 mit Service Pack 1. Weiter benötigt Simio das .NET Framework 4.6.2
Performance
Innerhalb der ersten Phase *Proof of Concept*, wurde die Lösung zum Klonen von Modellen gefunden. Diese basiert auf einem Simio-eigenen UICommand. Dadurch kann von aussen keinen Einfluss auf die Performance bzw. die Response Time dieses Vorgangs genommen werden.
Lizensierung und Version von Simio
Simio wird in verschiedenen Versionen[4] ausgeliefert. Eine Einschränkung besteht dabei darin, dass die lizenzfreie Simio Personal Edition keine Addins durch den Benutzer aktivieren lässt. Daher wird zur Nutzung eine Lizenz vorausgesetzt. Weiter muss die Simio-Version lizensiert und somit aktiviert sein. Ansonsten wird die Version auf die Simio Personal Edition abgestuft.

[3] Simio Product Information: https://www.simio.com/products/SupplementalProductInformation.pdf
[4] Simio Editionen im Überblick: https://www.simio.com/products/SimioProductMatrixCommercial.pdf

4.7 Abstraktion von Simio

Zu Beginn des Projektes lag der Fokus auf der Abstraktion des internen Aufbaus von Simio. Dieser Aufbau ist gibt Aufschluss über die einzelnen Zusammenhänge innerhalb von Simio und ist massgebend für die Implementierung. Ein einzelnes Simio-Projekt besteht aus einem oder verschiedenen Models. Ein Model kapselt einerseits die konkreten Simulationsobjekte (Server, Sink, Source, usw.), andererseits Experimente. Durch Experimente können verschiedene Konfigurationen und Parameter getestet werden. Ein Experiment ist in verschiedene Szenarien unterteilt. Ein solches Szenario stellt eine Parameterkonfiguration dar. Die Resultate eines Szenarios werden in Responses gekapselt. Diese Responseresultate werden anschliessend miteinander verglichen.

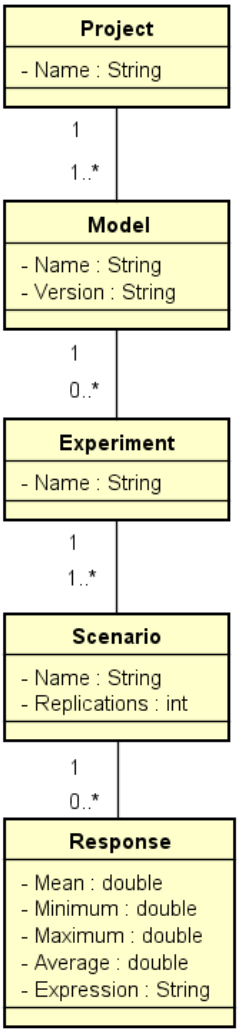


Abbildung 7: Aufbau eines Simio-Projektes

Innerhalb von Simio sind Experimente tabellarisch aufgebaut. Eine Response zieht sich jeweils als Spalte über alle Szenarien hinweg. Der in der Response enthaltene Endwert ist jeweils der Mittelwert aller Ergebnisse der Replikationen.

4.8 Konzept der Implementierung

Im Zentrum steht das Add-In ModelAnalyzer, welches Teilschritte des Modellierungsprozesses abbildet. Dieses teilt sich in mehrere funktionale Komponenten auf.

- Modellmanagement
- Modellvergleich

Das Modellmanagement setzt dabei die Erzeugung der Modellklone und Versionierung um. Die Vergleichskomponente übernimmt die Aufgabe des mathematischen Vergleichs.

Um die Daten von Simio abzufangen, wurde ein API-Adapter entwickelt, welcher den Sachverhalt, dargestellt in Abbildung 7, kapselt und entsprechend von der SimioAPI entkoppelt. Dieser soll als Plattform für weitere Projekte dienen.

Der ModelAnalyzer kann als Standalone-Applikation verwendet werden, welches es ermöglicht, auf bereits angelegte Simio-Projekte zuzugreifen und ohne die Verwendung von Simio Modellvergleiche durchzuführen. Zur Persistierung der Projekte wird eine SQLite-Datenbank [5] genutzt. Das Add-In selbst bildet die Funktionalität des Klonens und Vergleichens von Modellen ab. Weitere Features stellen die Visualisierung eines Vergleichs als Boxplot, das Exportieren von Vergleichsdaten in eine CSV-Datei und verschiedene Filteroptionen dar.

Zusätzlich ist ein Installer implementiert worden. Dieser unterstützt den Benutzer beim Installationsprozess von verschiedenen, an der HSR entwickelten, Simio-Add-Ins.

4.9 Use-Cases

Aus den Anforderungen sind die auf den nächsten Seiten aufgeführten Use-Cases hervorgegangen. Die Use-Cases sind nach den beiden Komponenten Modellmanagement und Modellvergleich aufgeteilt. Grün umrandete Use-Cases sind auch in der Standalone-Version verfügbar.

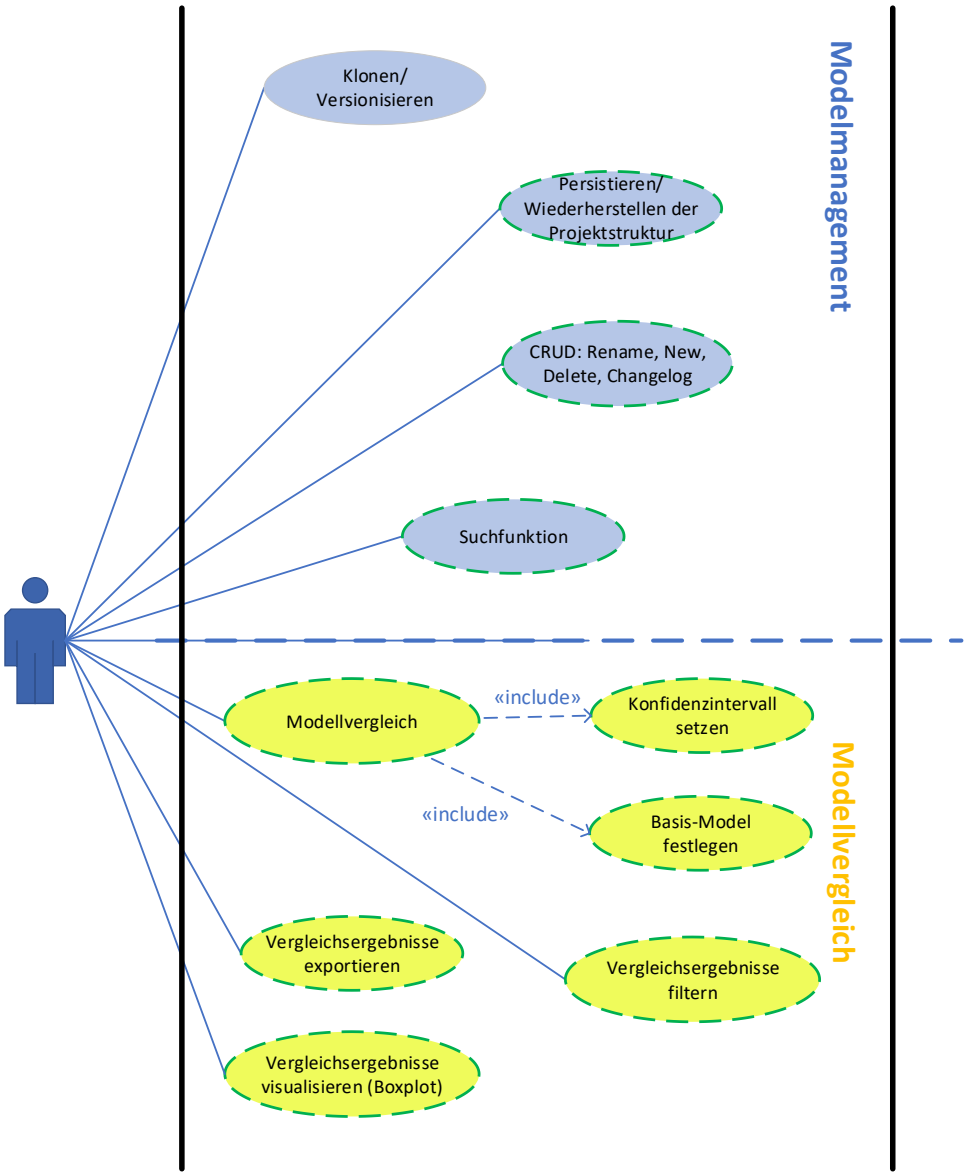


Abbildung 8: Use-Case Diagramm

4.9.1 Use-Cases: Modellmanagement

Use-Case: Klonen
Das Addin muss die bestehende Simio-Projektstruktur identisch wiedergeben. Der Benutzer ist in der Lage die bestehenden Modelle zu klonen. Der Benutzer legt den Namen und die Versionsnummer des Klons selber fest. Nach dem Klonvorgang wird der Klon einerseits im Add-In, wie auch in Simio dargestellt.

Use-Case: Persistieren/Wiederherstellen der Projektstruktur
Die im Add-In modifizierte Projektstruktur kann abgespeichert und beim nächsten Gebrauch wiederhergestellt werden. Die abgespeicherten Elemente umfassen folgende Eigenschaften/Felder: • Version • Changelog/Kommentar • Änderungen im Simio sollen automatisch in den ModelAnalyzer übernommen werden

Use-Case: Umbenennen, Neuerstellen, Löschen, Kommentieren/Changelog eines Modells
Der Benutzer kann ein Modell wie folgt über das Add-In modifizieren: • Umbenennen • Klonen • Version anpassen • Kommentieren/Changelog erfassen Nachdem die Operation abgeschlossen ist, sind die modifizierten Felder in Simio und im Addin aktualisiert.

Use-Case: Suchfunktion
Innerhalb des Modelmanagement-Bereichs wird eine Suchfunktion bereitgestellt. Dadurch kann der Benutzer die aktuelle Projektstruktur filtern und spezifisch nach Elementen suchen. Es kann nach folgenden Elementen gesucht werden: • Entities • Modelle • Experimente • Szenarios

4.9.2 Use-Cases: Modellvergleich

Use-Case: Modellvergleich

Durch das Selektieren von verschiedenen Modellen und/oder verschiedenen Szenarien können diese in die aktuelle Vergleichsansicht eingebunden werden. Die momentan selektierten Elemente können mit dem setzen der Basis miteinander verglichen werden. Ein solcher Vergleich rechnet die Abweichung immer von der gesetzten Basis her aus.

Includes:

- «Konfidenzintervall setzen»
- «Basis-Modell festlegen»

Use-Case: Konfidenzintervall

Über einen Regler kann der Benutzer das Konfidenzintervall (Signifikanzzahl) für den momentanen Vergleich einstellen.

Use-Case: Basis-Modell festlegen

Eingelesene Modelle und Experimente/Scenarien können über ein entsprechendes UI-Element als Basis festgelegt werden.

Use-Case: Vergleichsergebnisse strukturieren

Dem Benutzer werden verschiedene Möglichkeiten gegeben, die aktuellen Vergleichsergebnisse zu filtern. Der Filter umfasst folgende Optionen:

- Anzeige von: Signifikante Resultate mit positiver Abweichung
- Anzeige von: Signifikante Resultate mit negativer Abweichung
- Ausblenden von: Resultaten mit keinen signifikanten Abweichungen

Über ein separates Suchfeld kann nach bestimmten Elementen in der Vergleichsansicht gesucht werden. Die Suche beinhaltet folgende Elemente:

- Modell
- Experiment
- Szenario

Die Responses die aktuellen Modelle und Vergleich können über eine separate Liste ein- und ausgeblendet werden.

Use-Case: Vergleichsergebnisse visualisieren (Boxplot)

Die Vergleichsergebnisse können als Boxplot visualisiert werden. Diese Darstellung umfasst den momentan durchgeführten Vergleich und kann ebenfalls nach Responses gefiltert werden.

Use-Case: Exportieren

Der Benutzer kann die Ergebnisse des aktuellen Vergleichs in eine CSV-Datei zur Weiterverarbeitung exportieren.

5. ModelAnalyzer

Abbildung 9 zeigt das Gesamtsystem des Add-Ins und dessen Aufteilung in die verschiedenen Layer. Diese sind in den folgenden Kapiteln ausführlicher erklärt.

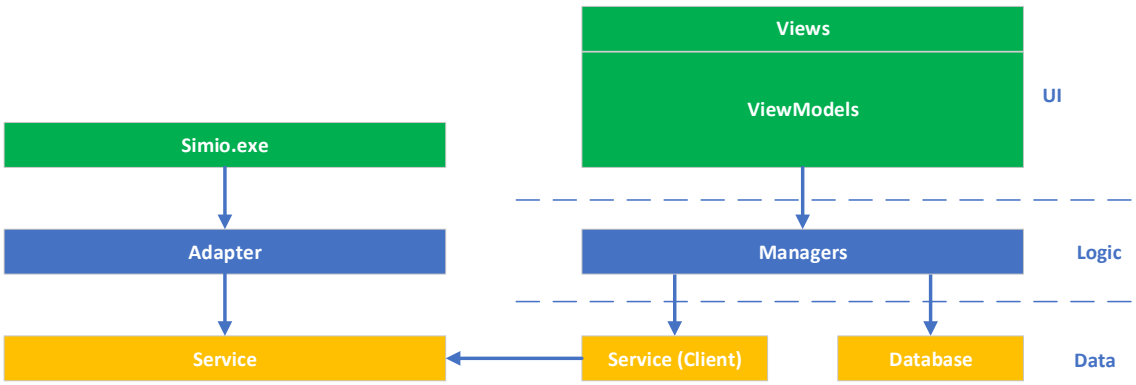


Abbildung 9: Layering

5.1.1 Physikalische Aufteilung

Dieses Projekt umfasst verschiedene Aspekte und Teillösungen. Daher hat es sich angeboten, diese in verschiedene Solutions aufzuteilen. Mit dieser Aufteilung sollen die einzelnen Komponenten getrennt voneinander nutz- und (wieder)verwendbar sein. Das Projekt ist in folgende Solutions aufgeteilt:

Solution	Kurzbeschreibung
Adapter.SimioAPI	Kapselung der SimioAPI in Form eines WCF-Services
Installer.AddinCollection	Installationsassistent
Addin.ModelAnalyzer	Add-In zum Modellvergleich

5.2 Layering

Das Add-In ist in verschiedene Layer unterteilt. Dieser Abschnitt erklärt die einzelnen Layer und deren Funktionalität und Verantwortlichkeiten. Spezifische Implementationen werden in späteren Kapitel erläutert.

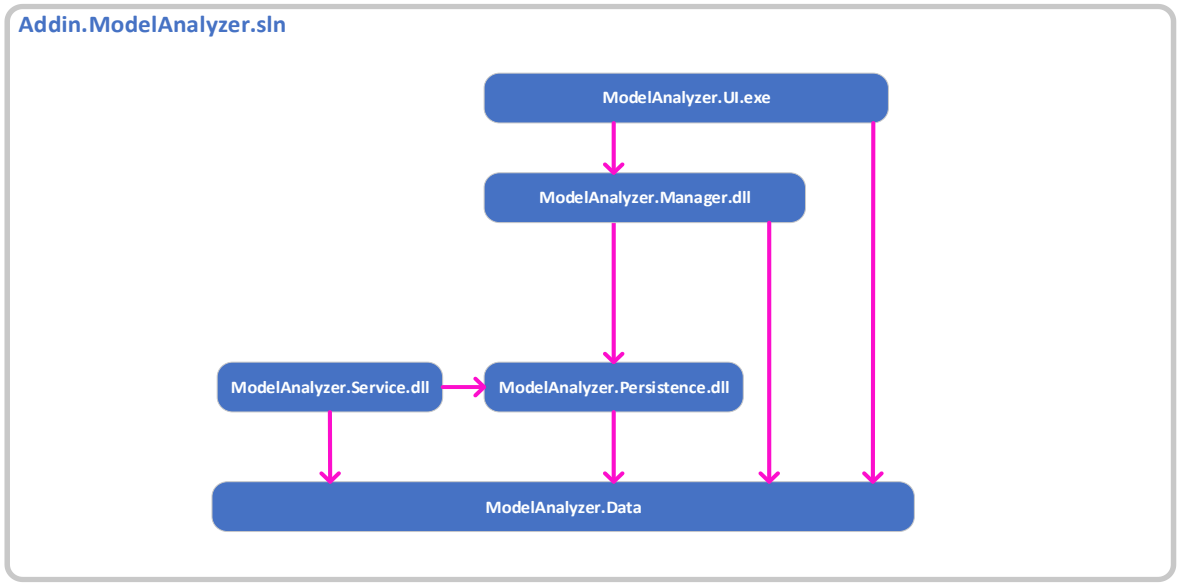


Abbildung 10: Gesamtübersicht Addin.ModelAnalyzer

5.3 UI-Layer – Addin.ModelAnalyzer.UI

Innerhalb des UI-Layers ist das graphische Benutzerinterface umgesetzt. Da es sich um eine WPF-Applikation (Windows Presentation Foundation) handelt, ist dieses nach dem MVVM-Pattern [6] (Model View ViewModel) aufgebaut. Für die einzelnen Elemente (Model, Entity, Scenario, Experiment) ist ein einheitliches ViewModel, *ProjectItemViewModel*, angelegt. Dieses wiederrum hält eine Liste mit *ResponseViewModel*s, welches intern ein Response-Objekt kapselt.

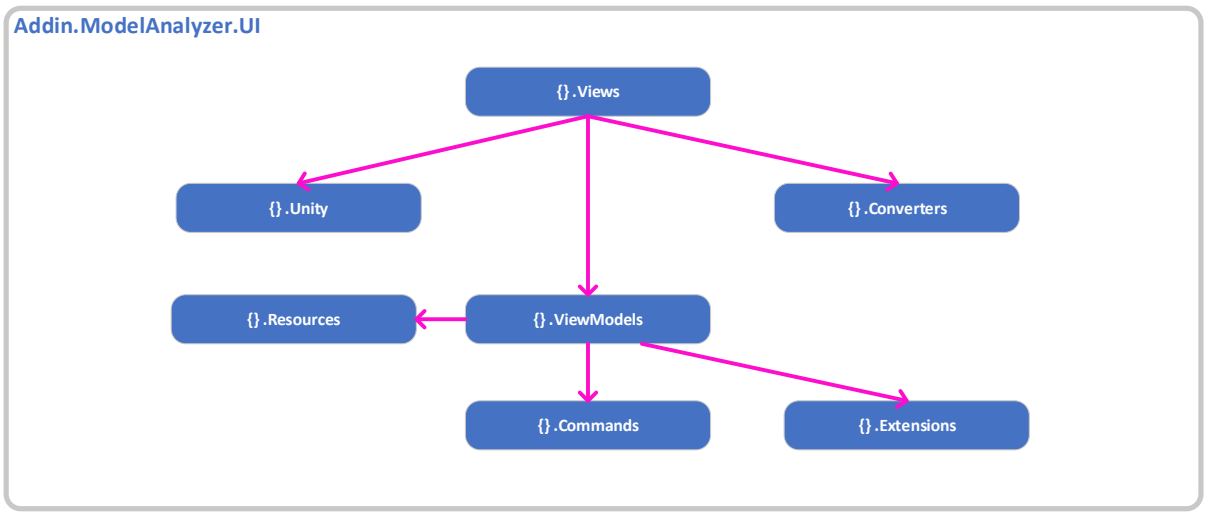


Abbildung 11: Namespaces des UI-Layers

5.4 Visualisierung

Simio stellt die Funktion der Visualisierung einzelner Responses bereits zur Verfügung. Jedoch ist diese Funktionalität nur auf einzelne Experimente limitiert. Ziel dieser Umsetzung ist es, modellübergreifende Boxplots zu erzeugen und somit eine graphische Aussage über den Modellvergleich zu treffen.

Die Implementierung findet mittels OxyPlot statt. Diese Library bietet die Möglichkeit Datensätze in verschiedenen Diagrammen zu visualisieren. Erzeugt werden die Boxplots im *VisualizationViewModel*. Die Darstellung wird in einem separaten Tab umgesetzt.

5.5 Logic – Addin.ModelAnalyzer.Managers

Die Logik des ModelAnalyzers ist im Logic-Layer angesiedelt. Dieser umfasst, getrennt nach Aufgabenbereich, verschiedene Manager. Die verschiedenen Befehle zum Verwalten und Vergleichen der Modelle werden hier zentralisiert zusammengeführt und auf verschiedene Managers aufgeteilt.

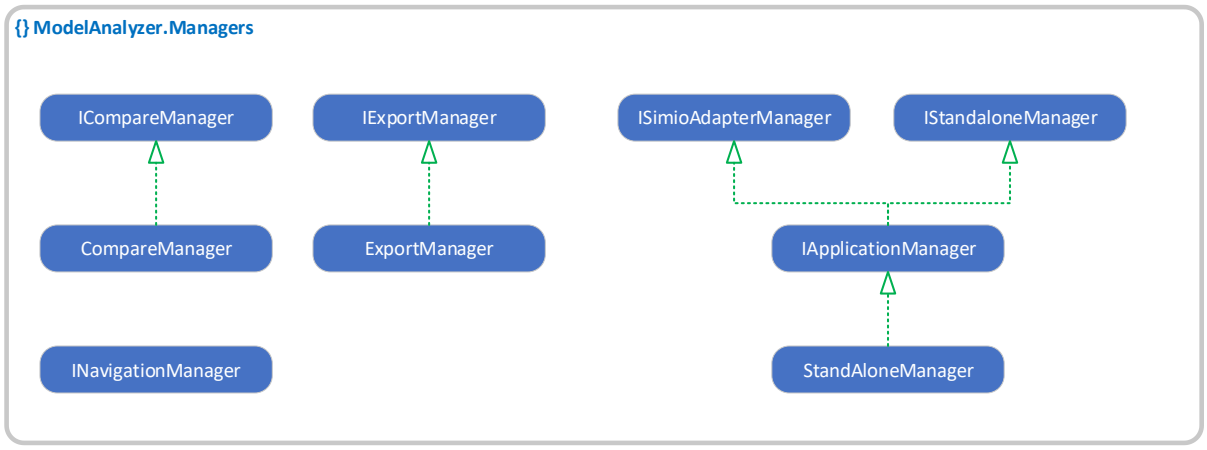


Abbildung 12: Übersicht Assembly Addin.ModelAnalyzer.Managers

ExportManager
Der *ExportManager* implementiert die Funktionalität des Exports der aktuellen Vergleichsergebnisse in eine CSV-Datei. Ziel dieser Funktion ist es, einfach an die verschiedenen Daten des Vergleichs zu gelangen und diese zur Weiterverarbeitung bereit zu stellen.

ApplicationManager
Der *ApplicationManager* steuert die gesamte Applikation. Beim Startvorgang wird unterschieden, ob die Applikation mit Simio bzw. einem Adapter-Service oder in der Standalone-Version läuft. Dies geschieht über das Auswerten der Event-Argumente. Falls diese Prozess-Argumente «simio» enthalten, wird die Applikation mit einem Adapter-Service gestartet. Andernfalls wird der Service durch einen *StandaloneManager* substituiert.

StandaloneManager

Falls die Applikation in der Standalone-Version gestartet wird, wird ebenfalls eine Instanz des *StandaloneManagers* erzeugt. Dieser übernimmt die Rolle als Platzhalter für den Adapter-Service. Durch ihn werden Daten aus der Datenbank geladen und interpretiert.

INavigationManager

Dieses Interface stellt die Methoden zur Erzeugung der verwendeten Dialogfenster zur Verfügung. Die konkrete Implementation liegt auf dem UI-Layer in Form des *MessageHandler*. Dieser nimmt Messages entgegen und löst daraufhin die entsprechenden Dialogfenster aus. Genutzt werden diese beispielsweise beim Klonen und Umbenennen eines Models.

CompareManager

Die Implementierung des tatsächlichen Modellvergleichs wird durch den *CompareManager* übernommen. Darin werden die mathematischen Vorgänge (siehe 3.8 Mathematische Grundlage des Modellvergleichs) abgebildet. Der Vergleich findet auf Stufe Response statt.

SimioUICommands

Die direkten Befehle an Simio, zum Verändern von effektiven Simio-Elementen, werden durch Simio-UICommands realisiert, welche durch die SimioAPI gegeben sind. Durch diese Commands sind folgende Operationen geregelt:

- Klonen
- Umbenennen von Modellen
- Aktualisieren der Projektstruktur

Innerhalb dieses Layers werden diese Befehle vom UI entgegengenommen und entsprechend and das Service-Gegenstück (Proxy) weitergeleitet.

Versionierung

Jedes Model, Experiment und Szenario wird mit einer Versionsnummer versehen. Falls eine neue Projektstruktur eingelesen wird, so werden die Elemente mit der Version 1.0 initialisiert. Während des Klonvorgangs wird der Benutzer über einen entsprechenden Dialog aufgefordert, den Namen und die Version des zu erzeugenden Klons festzulegen. Somit kann der Benutzer das Namens- und Versionierungsschema frei wählen.

Speichern / GUID / interne IDs

Simio nutzt zur Identifizierung der einzelnen Projekte und deren Elemente GUIDs. Jedoch werden einem Projekt untergeordnete Elemente mit derselben GUID bestückt. Durch diese Praktik werden die einzelnen Elemente nicht mehr eindeutig über diese GUIDs identifiziert. Dieser Aspekt der Funktionsweise von Simio stellt sich als wesentlich für die Persisiterung der einzelnen Elemente in der Datenbank des Addins heraus. Um Mehrdeutigkeiten dieser Identifier zu vermeiden, wird innerhalb des Data-Layers jedem einzelnen Element eine neue ID vergeben. Diese dient während des Wiederherstellungsprozesses zum Mapping der einzelnen Datensätze in der Datenbank mit den Gegenstücken innerhalb der Simio-Projektstruktur.

5.6 Data – Addin.ModelAnalyzer.Data

Der Data-Layer gilt als zentraler Zugriffspunkt für alle Modeldaten. Jede andere Schicht, welche spezifische Modeldaten benötigt, kann diese vom Data-Layer beziehen. Diese Art der Umsetzung verringert die Duplizierung von Code und Objekten. Zusätzlich muss nicht auf jeder anderen Schicht ein Mapping durchgeführt werden. Weiter findet hier das Mapping zur Datenbank statt. Als OR-Mapper wird SQLite eingesetzt. Die einzelnen Objekte werden durch die entsprechenden SQLite-Annotations auf die darunterliegenden Tabellen und Spalten gemappt.

Durch den Enum *ProjectItemType* werden die einzelnen *ProjectItems* in ihre Rollen überführt. So können leicht weitere Simio-Elemente abgebildet und dem System bekannt gemacht werden.

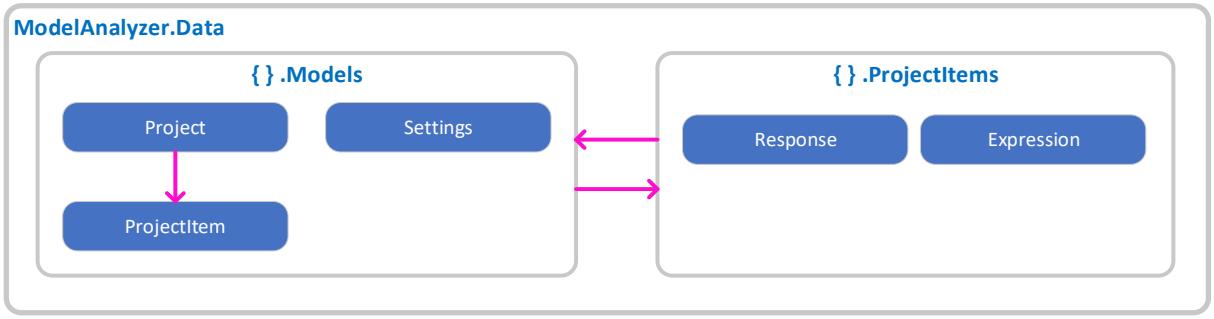


Abbildung 13: Übersicht Assembly Addin.ModelAnalyzer.Data

5.7 Persistence – Addin.ModelAnalyzer.Persistence

Dieser Layer realisiert die Datenbank-Operationen. Als Datenbank ist SQLite [5] eingesetzt. Innerhalb der Implementation von *IDatabase* sind die schreibenden und lesenden Operationen gegen die Datenbank umgesetzt.

Im Betrieb mit Simio können Versionen und Kommentare erfasst werden. Diese werden innerhalb der Datenbank gespeichert. Bei erneuten Bearbeiten eines Simio-Projekts, werden die entsprechenden Datensätze gelesen und im ModelAnalyzer erneut zugeordnet. Alle Operationen gegen die Datenbank sind asynchron umgesetzt. Dies umfasst auch die Wahl der Datenbank, SQLite bietet zwei verschiedene Produkte an. Durch diese asynchrone Umsetzung blockieren Operationen der unteren Layer den UI-Thread nicht.

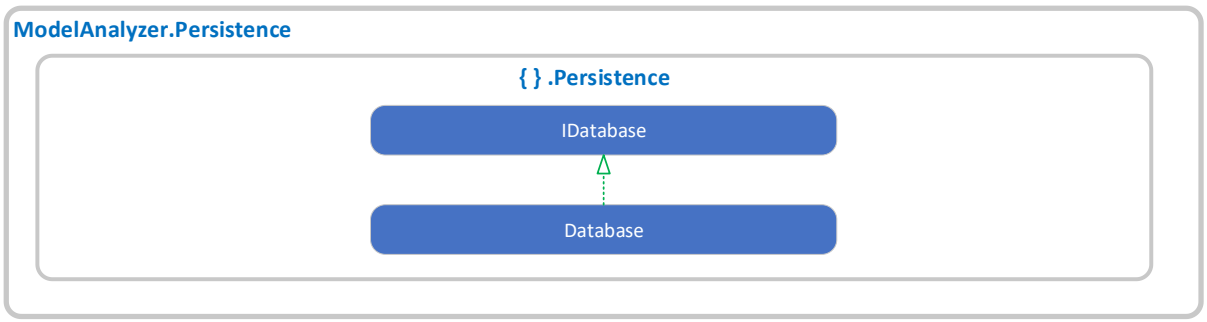


Abbildung 14: Übersicht Assembly Addin.ModelAnalyzer.Persistence

5.8 Prozesse und Laufzeitvarianten

Der ModelAnalyzer wird immer in einem eigenständigen Prozess ausgeführt. Zur Interprozesskommunikation mit Simio wird WCF [7] genutzt. Durch die Implementierung als Add-In und als Standalone-Variante wird daher auch zwischen zwei Laufzeitvarianten unterschieden. Der Unterschied der beiden Varianten liegt dabei in der Art und Weise, woher die Daten bezogen werden. Zusätzlich unterliegt die Standalone-Variante funktionalen Einschränkungen.

5.8.1 ModelAnalyzer als Simio-Add-In

Wird der ModelAnalyzer mit Simio verwendet, stehen alle Funktionen zur Verfügung. Als Einstiegspunkt zum Start der Applikation dient das entsprechende *IDesignAddin*. Durch dessen *Execute()*-Methode wird eine neue Instanz des API-Adapters erzeugt, welcher wiederrum den ModelAnalyzer-Prozess erzeugt.

5.8.2 Standalone

Der ModelAnalyzer ist als Standalone-Version verfügbar. Wird die Applikation in diesem Modus gestartet, wird die Funktionalität eingeschränkt, jedoch kann sie ohne eine laufende Instanz von Simio genutzt werden. Der Standalone-ModelAnalyzer enthält folgende Funktionalität:

- Vergleich von Modellen
- Export von Vergleichsresultaten
- Visualisierung von Modellvergleichen
- Filteroptionen

Als einzige Rahmenbedingung zum effektiven Gebrauch des Standalone-ModelAnalyzer gilt der Umstand, dass vorher Daten in die Datenbank abgelegt worden sind.

Der Unterschied in der Startsequenz liegt darin, dass die Simio-Instanz und somit der *DesignContext* wegfällt. Hier wird als Einstiegspunkt direkt *App.xaml.cs* genutzt. Durch Überprüfung der Event-Argumente kann so entschieden werden, ob Simio bzw. ein *SimioAdapter* läuft. Entsprechend wird durch Unity [8] eine andere Container-Konfiguration initialisiert.

6. Adapter.SimioAPI

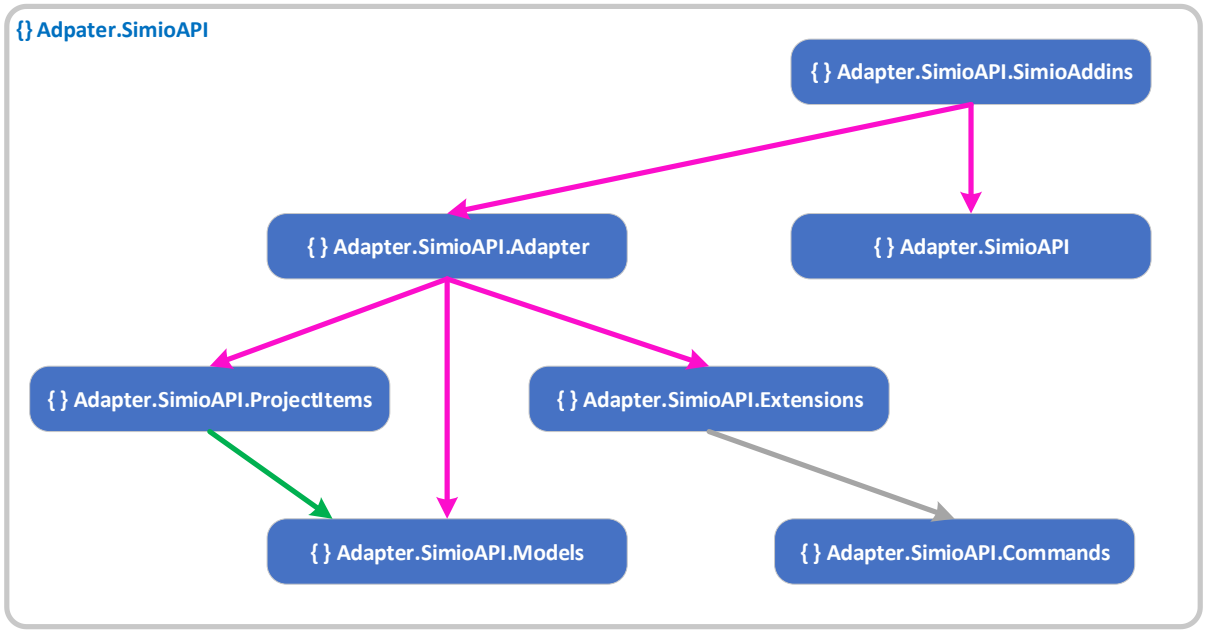


Abbildung 15: Übersicht Adpater.SimioAPI

6.1 Grundgedanke

Mit der Umsetzung des *Adapter.SimioAPI* werden verschiedene Ziele verfolgt. Einerseits ist dadurch eine Entkopplung des Addin und der SimioAPI gelungen. Andererseits ist durch diesen Adapter die Möglichkeit auf einfache Erweiterbarkeit und Wiederverwendbarkeit von Code gegeben. Dieser Abschnitt gibt Aufschluss über die Umsetzung und die Funktionsweise der Kapselung.

Die Erweiterbarkeit sowie die Wiederverwendbarkeit wird dadurch gewährleistet, dass Teile der SimioAPI in einem ersten Schritt gekapselt und anschliessend als WCF-Service zur Verfügung gestellt werden. Abbildung 16 zeigt diesen Sachverhalt auf. Somit kann der *Adapter.SimioAPI* als Plattform für weitere Projekte und Arbeiten eingesetzt werden.

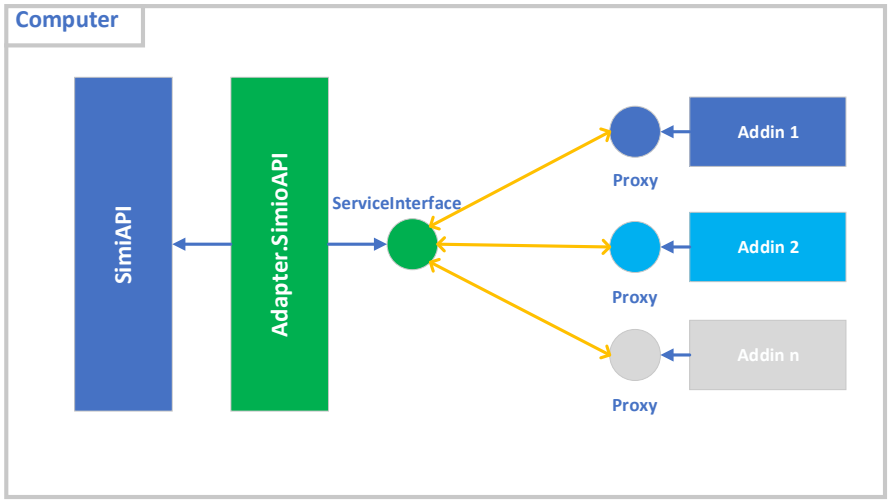


Abbildung 16: Adapter.SimioAPI – Übersicht

Dadurch, dass der Adapter der SimioAPI zentral als Service angeboten wird, kann jedes Add-In simultan durch entsprechende Interfaces Daten beziehen. Der Adapter besteht aus einem Service-Interface, dieses definiert die *OperationContracts* und einer entsprechenden Implementation. Als Instanzmodell ist *SessionMode.Required* gesetzt.

Die Erweiterbarkeit ist dadurch gewährleistet, dass der Service lediglich um die gewünschten Methoden und somit Simio-Daten erweitert werden kann. Diese sind anschliessend für alle Add-Ins, welche als Endpoint fungieren, verfügbar.

6.2 Interprozesskommunikation

ModelAnalyzer läuft in einem eigenen Prozess. Dieser wird beim Starten des Add-Ins erzeugt. Zur Interprozesskommunikation dient ebenfalls WCF. Durch ein von WCF gegebenes *NetNamedPipeBinding* wird ein Runtime-Callstack zwischen zwei oder mehreren WCF-Endpoints (Proxy) aufgebaut. Dieses Binding ist speziell für on-machine Interprozesskommunikation optimiert. Implementiert ist dies auf der Client- bzw. Add-In-Seite, innerhalb des Service-Layers.

Die Implementation des Service-Interfaces beherbergt einerseits Methoden um auf Simio-Projektdaten zuzugreifen, andererseits werden darüber auch Commands, welche im Simio ausgeführt werden sollen, abgesetzt. Diese Commands sind durch die SimioAPI bereits gegeben (*SimioUICommands*) und werden über den *DesignContext* abgesetzt, welcher durch den Adapter ebenfalls gekapselt wird.

6.3 Kapselung von IModel und IDesignContext

Durch den Adapter werden einzelne Teile der SimioAPI gekapselt. Somit wird eine Entkopplung dieser API erreicht. Zurzeit werden lediglich Daten, welche für die Umsetzung des ModelAnalyzers nötig sind gekapselt, eine Erweiterung ist jedoch leicht erreichbar. Die Kapslung umschliesst Teile von *IDesignContext* und *IModel*. Um diese vollständig von der SimioAPI zu lösen, wurden Äquivalenzklassen analog der gekapselten Daten erzeugt. Die Äquivalenzklassen umfassen eine vereinfachte, auf den ModelAnalyzer zugeschnittene Struktur, welche grob eine standardmässige Simio-Projektstruktur abbildet.

Die einzelnen Properties dieser Klassen sind mit den WCF-Annotationen entsprechend als *DataMemeber* bezeichnet.

Das teilweise gekapselte Interface *IModel* enthält alle Daten, welche spezifisch mit der Modellstruktur eines Simio-Models zusammenhängen. Dies umschliesst einerseits den in Abbildung 7 dargestellten Sachverhalt, andererseits Elemente, Schedules, Tabellen und Properties. *IModel* operiert lediglich auf der Stufe Modell.

IDesignContext hingegen setzt eine Stufe über *IModel* auf. Im DesignContext sind die aktuell aktiven Elemente der Typen *IModel*, *IExperiment* und *ISimioProject* enthalten. Der DesignContext ist zusätzlich ein wesentlicher Bestandteil, weil durch ihn *SimioUICommands* abgesetzt werden können.

Die effektive Umsetzung der zu kapselnden Daten zeigt das Klassendiagramm in Abbildung 17 auf. Gestützt auf die Basisklasse ProjectItem, sind die einzelnen Simio-Elemente als ableitende Klassen modelliert.

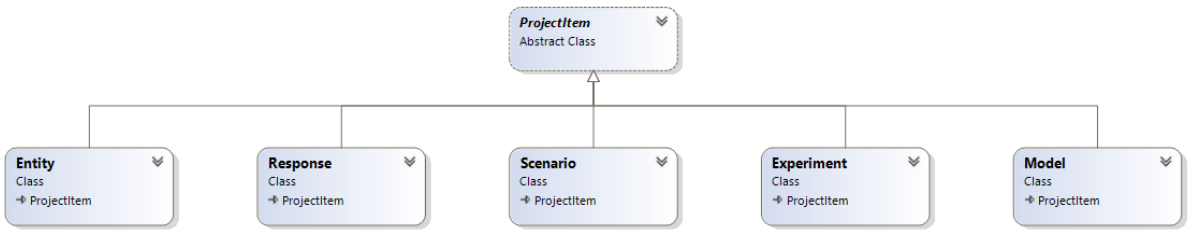


Abbildung 17: Klassendiagramm - Umsetzung der Kapselung

6.3.1 Simio Add-Ins

Weiter sind innerhalb von *Adapter.SimioAPI* ebenfalls die einzelnen UserAddins abgelegt. Simio nutzt als Einstiegspunkt eines Addins eine Klasse, welche vom Simio-Interface *IDesignAddin* erbt. Für jedes Addin muss eine solche Klasse existieren. Um den Grundgedanken der losen Kopplung möglichst vollständig umzusetzen, ist die Entscheidung gefallen, diese innerhalb von Adapter.SimioAPI unterzubringen.

Einschub: IDesignAddin

Simio liefert im Zuge seiner API vorgefertigte Visual Studio Templates mit, welche das Grundgerüst eines Addins vorgeben. Dieses Klassenskelett besteht aus Properties (Name, Description, Icon, usw.) und aus den beiden Methoden *Execute(IDesignContext context)* und *Shutdown()*. Innerhalb dieser Methoden wird das entsprechende Addin gestartet bzw. beendet.

NFR: Extensibility

Durch Adapter.SimioAPI wird die nichtfunktionale Anforderung Extensibility (Erweiterbarkeit) abgedeckt. Folgeprojekten ist es nun möglich auf leichte und effiziente Weise Simio um weitere Addins zu erweitern. Ein neues Addin kann leicht und völlig losgelöst von Simio entwickelt werden. Anschliessend muss lediglich eine neue Addin-Klasse erzeugt eingebunden werden. Falls weitere Daten von Simio selbst benötigt werden, kann das Service-Interface leicht um diese erweitert werden.

NFR: Reusability

Ebenfalls abgedeckt wird die NFR Reusability (Wiederverwendbarkeit). Dies wird durch den Umstand erreicht, dass Adapter.SimioAPI einerseits von allen, auf dem Computer laufenden, Addins genutzt werden kann, andererseits muss nicht für jedes Addin eine eigene Wrapper-Klasse erzeugt werden.

7. Simio Tools Installer

Im Rahmen dieses Projekts ist zusätzlich ein eigenständiger Installer entwickelt worden. Durch den Simio Tools Installer soll der Benutzer einfach und leicht von der HSR entwickelte Simio-Addins installieren können. Der Simio Tools Installer basiert auf dem WIX Toolset [9]. Dieses bietet eine vollumfängliche Library, welche es Entwicklern ermöglicht, auf einfache Art und Weise .msi-Dateien zu erzeugen.

Durch die Nutzung des WIX-Toolset kann eine graphische Benutzeroberfläche erzeugt werden. Diese setzen jedoch eine strikte Einhaltung von verschiedenen Sequenzen auf den Berechtigungsebnen «Benutzer» und «Administrator» voraus. Durch diese Einschränkung müssen einige Dialoge zwangsweise angelegt werden, obwohl sie momentan nicht alle genutzt werden.

AdminUISequence

- FatalErrorAdmin
- AdminExit
- ExitDialog
- PrepareDialog
- AdminWelcomeDialog
- ProgressDialog

InstallUISequence

- FatalError
- UserExit
- ExitDialog
- PrepareDialog
- WelcomeDialog
- ResumeDialog
- ProgressDialog

Durch diese Sequenzen werden wiederrum die verschiedenen Funktionalitäten zur Verfügung gestellt.

7.1.1 Installationssequenz

Die untenstehende Grafik zeigt die durch den Installer festgelegte Installationssequenz.

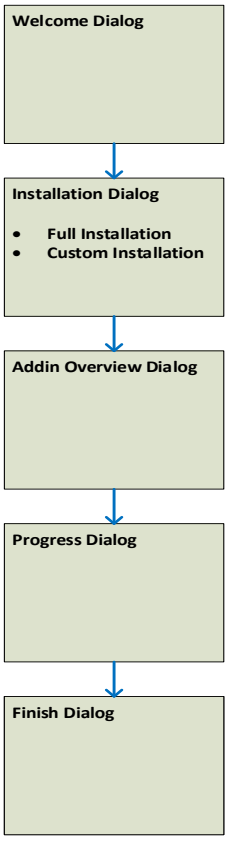


Abbildung 18: Installationssequenz

7.1.2 Modify-, Repair-, Removesequenz

Die folgende Grafik zeigt verschiedene weitere Sequenzen auf. Ausgelöst werden diese durch ein erneutes Ausführen des Simio Tools Installers oder über die Systemsteuerung des Computers.

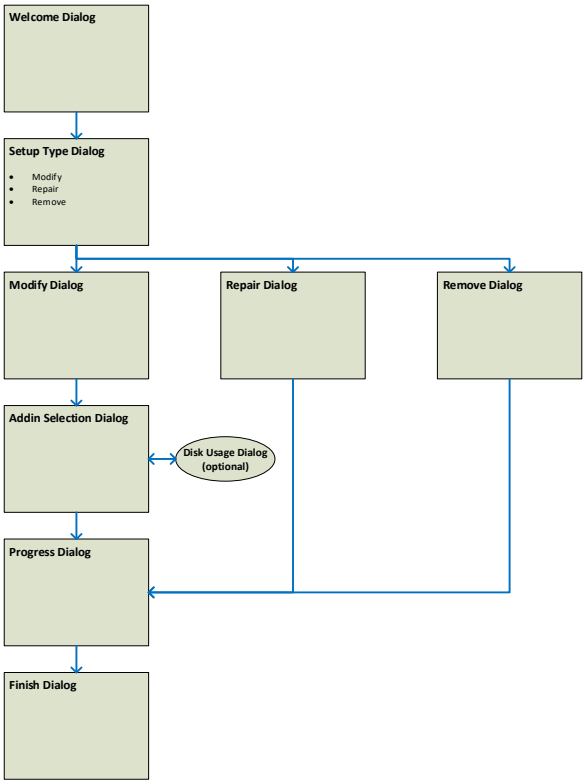


Abbildung 19: Modifizierungs- Reparatur- und Deinstallationssequenz

Durch diese erweiterte Funktionalität können folgende Aufgaben durchgeführt werden:

Modifikation (Modify- Flow)
Durch diese Sequenz können weitere Addins hinzugefügt oder entfernt werden.

Reparatur (Repair- Flow)
Diese Sequenz repariert die installierten Addins. Dies erfolgt über eine Deinstallation und erneuter Installation aller Addins.

Deinstallation (Remove-Flow)
Durch diese Sequenz werden alle Addins sowie der Simio Tools Installer vom Computer entfernt. Falls nur einzelne Addins entfernt werden sollen, muss die Modify-Sequenz genutzt werden.

7.1.3 Umsetzung

Die innere Struktur des Installers wird vom WIX Toolset vorgegeben. So befindet sich in der Datei «Product.wxs» der gesamte Umfang. Die XML-Datei teilt sich in zwei Bereiche auf: Komponenten und UI. Innerhalb des Komponenten-Bereichs werden alle benötigten DLLs aufgeführt und als «Components» markiert. Diese umfassen zum einen alle DLLs des Add-Ins, sowie externe Libraries. Weiter werden in dieser Sektion die Standardinstallationspfade gesetzt. Der UI-Bereich beherbergt alle Dialoge und deren Sequenzen. Sämtliche UI-bezogene Ressourcen, wie zum Beispiel Bilder und Icons, werden hier referenziert.

NFR: Deployment
Durch den Simio Tools Installer kann die nicht-funktionale Anforderung «Deployment» abgedeckt werden. Ein wichtiger Aspekt liegt darin, dass der Benutzer angenehm über die graphische Benutzeroberfläche Add-Ins installieren kann.

NFR: Extensibility
Die NFR «Extensibility» wird ebenfalls durch den Simio Tools Installer abgedeckt. Weitere Add-Ins können leicht als neue Komponente innerhalb der Component-Section eingefügt werden. Dadurch, dass die graphische Benutzeroberfläche bereits vollumfänglich besteht, wird das neue Add-In direkt integriert.

8. UX

Dieser Abschnitt zeigt die einzelnen Views des ModelAnalyzers. Es wird auf die beiden durchlaufenen UI-Konzeptionsphasen eingegangen und die darin entwickelten Resultate aufgezeigt.

8.1 Konzept

Die Entwicklung des GUI (Graphical User Interface) ist in zwei konzeptionelle Phasen unterteilt. Diese decken sich mit den vorher definierten Projektphasen. Die erst Phase beinhaltet die Erarbeitung eines Low-Fi Prototyps mittels Wireframes und die grobe Umsetzung innerhalb eines Prototyps. Innerhalb der zweiten Phase wurden die einzelnen Views während deren Implementierung verfeinert. Die Implementierung teilte sich in verschiedene Sprints auf. Jeder Sprint umfasste definierte Views und Funktionalitäten. Innerhalb eines solchen Sprints sind die Views dann in eine finale Form gebracht worden.

Als grundsätzlicher Leitfaden zur Entwicklung des UI, fiel der Entscheid, eine möglichst an Simio angelehnte Benutzeroberfläche zu erzeugen. Dadurch kann wird ein Grundverständnis und Wiedererkennungswert für den Benutzer geschaffen und der Einstieg enorm erleichtert.

8.1.1 HSR-Tab

Durch die vorangegangene Studienarbeit wurden bereits Tools und Add-Ins für Simio entwickelt. Mit diesen Voraussetzungen haben wir entschieden, dass eine Sammlung von HSR-Simio-Tools in Form eines separaten Tabs innerhalb von Simio als Container dienen soll. Für jedes spezifische Tool wird ein Button innerhalb dieses Tabs implementiert, über welche das entsprechende Add-In gestartet werden kann. Momentan wird lediglich der ModelAnalyzer eingebunden, sowie ein Button, welcher auf die Simulations-Webseite der HSR verweist.

8.2 Wireframes

Durch die Unterteilung der Funktionalität in zwei Komponenten (Modelmanagement und Modellvergleich), dieser Gedanke ist auch in den ersten Prototyp eingeflossen. Das Add-In soll in einem eigenständigen Fenster geöffnet werden.

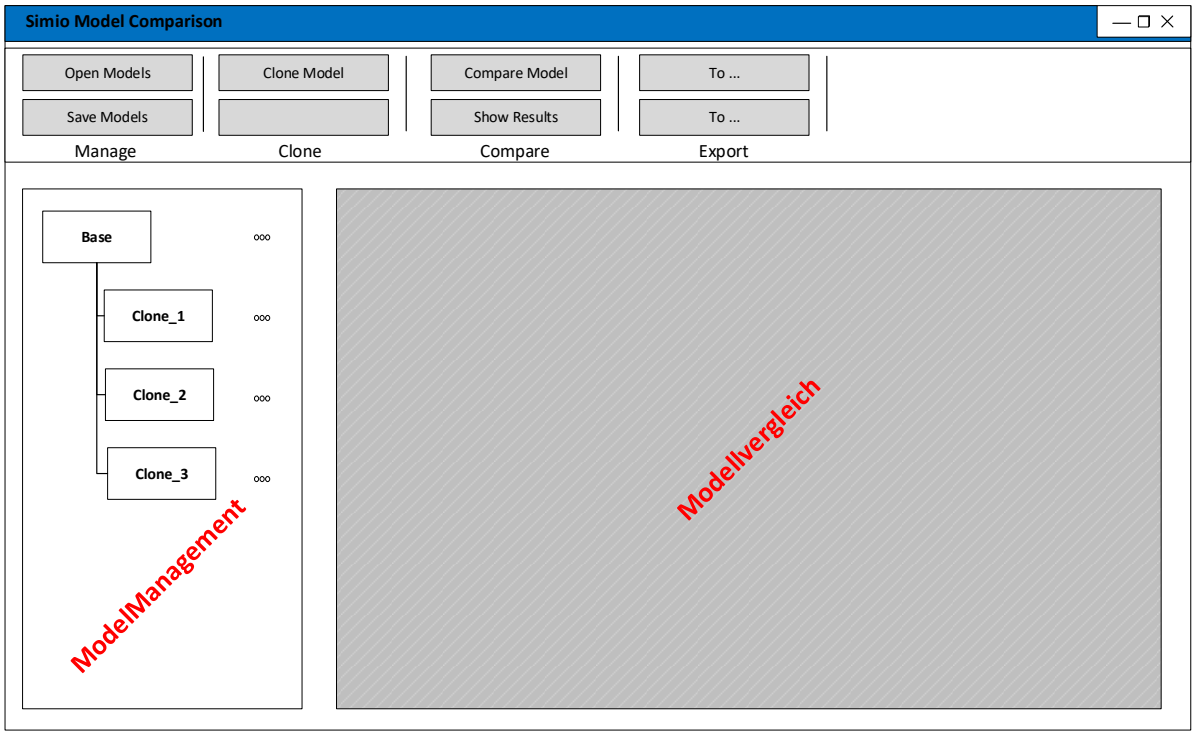


Abbildung 20: Mainview - Wireframe

Die Auflistung der verschiedenen Models soll als Baumstruktur realisiert sein. Dabei können sie auf- und eingeklappt werden, was die untergeordneten Elemente wie Szenarios und Experimente sichtbar macht.

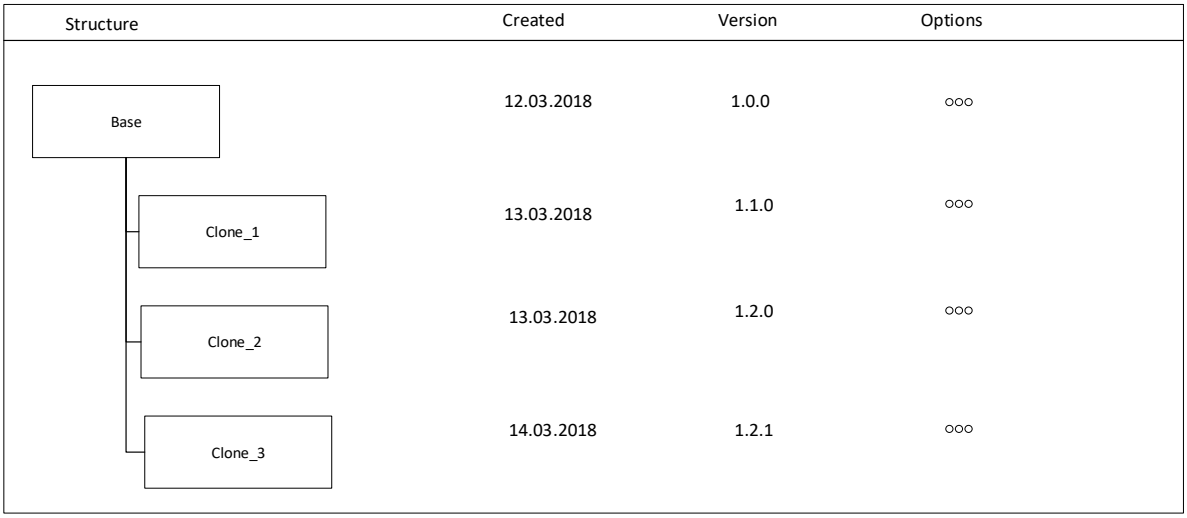


Abbildung 21: Clone-/ Strukturview Wireframe

Der erste Entwurf der Vergleichsansicht stellt die angewählten Modelle und deren Experimente erneut in Baumform dar. Dabei sollen Responses, welche die gleiche Expression aufweisen, auf derselben Zeile stehen und so den Vergleich visuell unterstützen.

Base	Clone_1	Clone_2
▸ Experiment 1	▸ Experiment 1	▸ Experiment 1
▾ Experiment 2	▾ Experiment 2	▾ Experiment 2
Response 1 Response 2	Response 1 Response 2	Response 1 Response 2
Scenario 1 ResultValue ResultValue	Scenario 1 ResultValue ResultValue	Scenario 1 ResultValue ResultValue
Scenario 2 ResultValue ResultValue	Scenario 2 ResultValue ResultValue	Scenario 2 ResultValue ResultValue
▸ Experiment 3	▸ Experiment 3	▸ Experiment 3

Abbildung 22: Resultview – Wireframe

8.3 Umsetzung

Das Add-In besteht aus einem Fenster, welches sich in zwei Subansichten unterteilt: **Modellmanagement** und **Modellvergleich**. Am oberen Rand befindet sich eine Ribbon-Taskbar.

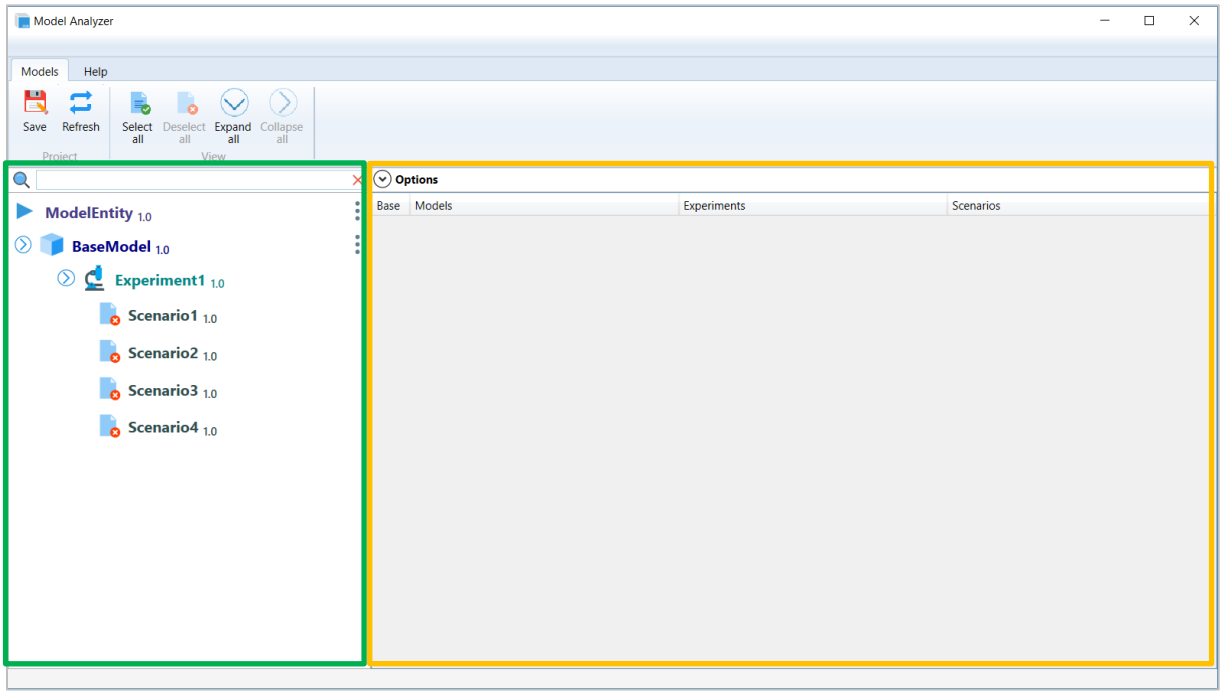


Abbildung 23: Mainview ModelAnalyzer

Ribbon / Taskleiste	
Icon/Symbol	**Beschreibung**
	Speichern der aktuellen Projektstruktur
	Refresh/Aktualisieren der Projektstruktur
	An- /Abwählen aller Elemente der Projektstruktur
	Ein- /Ausklappen aller Modelle und Experimente

8.3.1 Modellmanagement

Dieser Bereich stellt die Funktionalitäten zum Modellmanagement zur Verfügung. Angezeigt werden alle Projektelemente innerhalb des aktuellen Simo-Projektes. Dabei sind den Modellen immer die dazugehörigen Experimente und Szenarien untergeordnet, welche durch das **Pfeilsymbol** ein- und ausgeblendet werden können. Die Verwaltungsoptionen eines Modells können durch das Anwählen des **Optionsbuttons** eingeblendet werden. Über das **Suchfeld** kann nach einzelnen Elementen (Modell, Experiment, Szenario) gesucht werden.

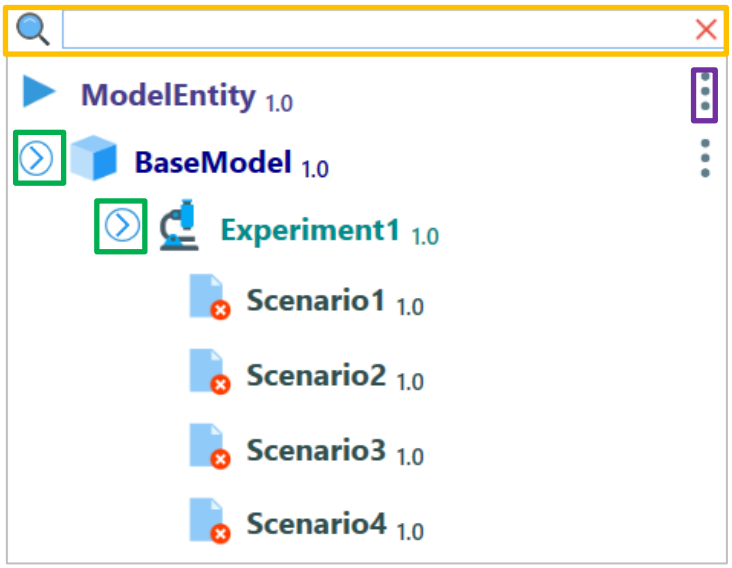
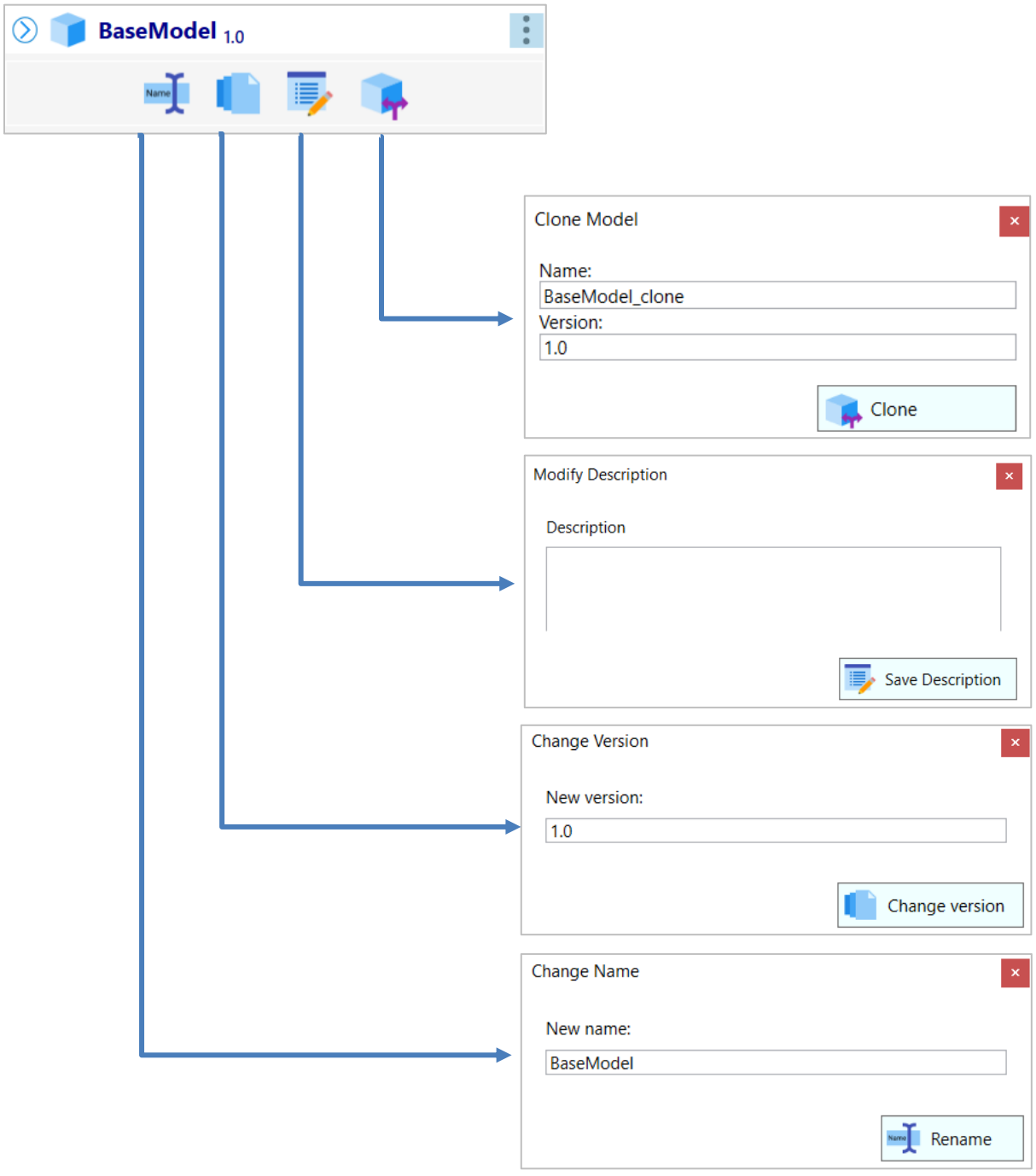


Abbildung 24: Modelmanagement – View

Die Verwaltungsoptionen werden anschliessend unterhalb des jeweiligen Modells eingeblendet. Die Optionen sind als Icons dargestellt. Alle Icons sind mit einem Tooltip hinterlegt. Durch das Auswählen einer Operation wird ein Dialogfenster zur Erfassung der Daten geöffnet. Sobald der jeweilige Dialog bestätigt wurde, findet die Operation statt.



Modellmanagement			
Icon/Symbol	**Beschreibung**	**Icon/Symbol**	**Beschreibung**
	Entity		Umbenennen
	Model		Version ändern
	Experiment		Kommentieren
	Szenario (An- / Abgewählt)		Klonen

Durch das Selektieren eines Szenarios werden diese im Vergleichsbereich angezeigt. Die Spalten und Zeilen erweitern sich dynamisch mit der Selektion. Selektierte Elemente sind anschliessend blau hinterlegt.

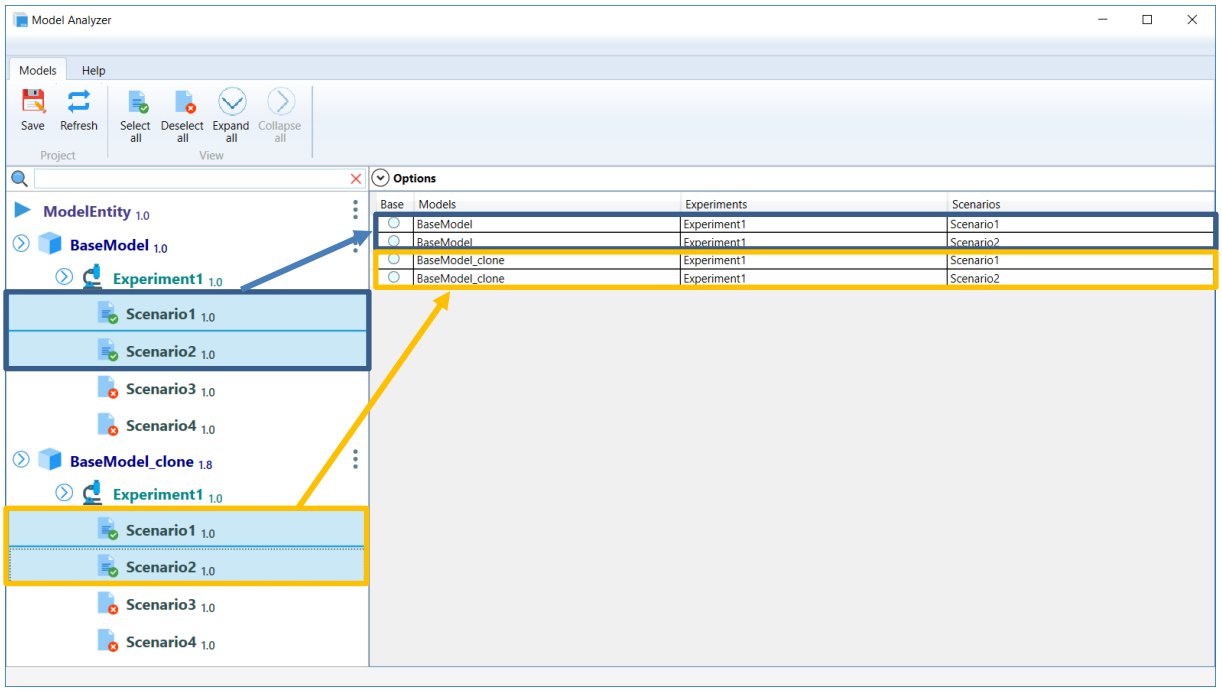


Abbildung 25: Darstellung der Multiselektion

8.3.2 Modellvergleich

Der zweite Bereich stellt die zu vergleichenden Modelle dar. Um den Benutzer möglichst an Simio zu erinnern, sind die Szenarien in tabellarischer Form aufgelistet. Die einzelnen Zeilen repräsentieren jeweils ein ausgewähltes Szenario aus der Modellmanagement-Ansicht. Die erste Spalte beinhaltet Radio-Buttons. Durch das Setzen eines solchen, wird das Basis-Modell bestummen. Die Zeile, welche das aktuelle Basis-Modell enthält, ist jeweils gelb hinterlegt. Der Vergleich wird automatisch durch das Setzen einer Basis ausgelöst.

Der Vergleich der einzelnen Modelle ist durch Farben visuell unterstützt. So sind signifikante Abweichungen mit einer negativen Werteabweichung rot, mit positiver grün und mit keiner weiss hinterlegt. Eine entsprechende Legende ist als Tooltip realisiert, welche durch das Darüberfahren eines Resultats angezeigt wird.

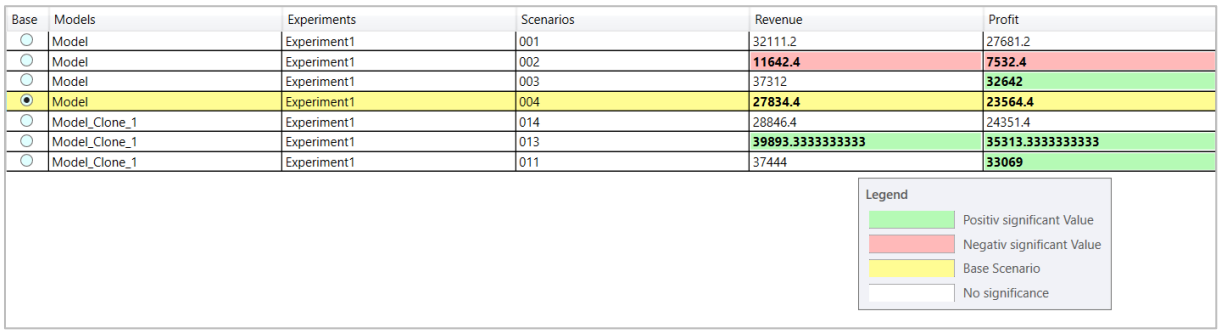


Abbildung 26: Vergleichsansicht

Die einzelnen Spalten zeigen die Namen der Modelle, Experimente und Szenarien. Für jede Response eines Szenarios wird dynamisch eine neue Spalte generiert. Die Spalten-Header dienen zur Sortierung, welche durch einen Klick von auf- nach absteigend geändert werden kann.

Die aktuellen Vergleichsdaten können durch verschiedene Optionen strukturiert werden. Diese Vergleichsoptionen befinden sich in einem ein- /ausklappbaren Bereich, welcher in drei Subgruppen gegliedert ist. Durch die Optionen in der Gruppe «Filter» können die Vergleichsergenisse nur auf solche mit signifikanter Abweichung eingegrenzt werden. Weiter kann zwischen postiver und negativer Abweichung gefiltert werden.

Die zweite Gruppe gibt die Möglichkeit, den Konfidenzintervall des Vergleichs festzulegen. Diese Einstellung hat Einfluss auf die Signifikanzzahl.

Innerhalb der Gruppe «Select Expression» können einzelne Responses ein- bzw. ausgeblendet werden. Die Spalten werden dynamisch nachgeladen.

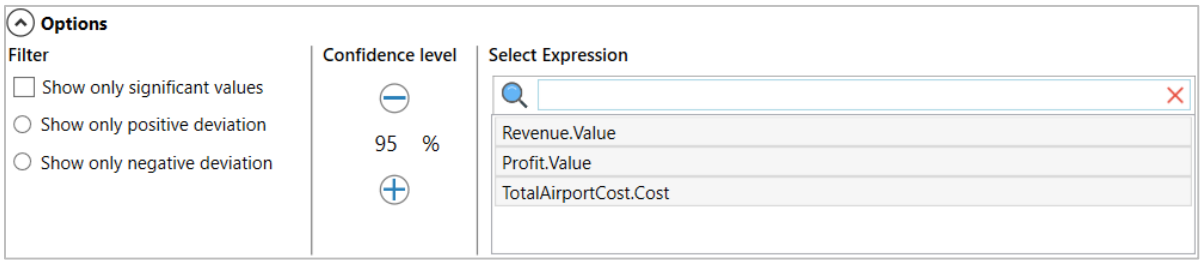


Abbildung 27: Vergleichsoptionen

8.3.3 Visualisierung als Boxplot

Die Visualisierung der verglichenen Responses findet im Vergleichsbereich statt. Dazu wurde am unteren Ende dieses Bereichs ein neuer Tab eingeführt. Diese Darstellung lehnt sich an die Visualisierung innerhalb von Simio an.

Einschub: Aufbau eines Boxplots

Durch einen Boxplot lassen sich Stichprobendaten übersichtlich und informativ darstellen. Ein solcher Boxplot besteht aus verschiedenen Teilelementen. Der Rumpf eines Boxplot besteht immer aus einem Rechteck und zwei Linien, welche den Rumpf verlängern. Das Rechteck repräsentiert dabei den Bereich, in welchem 50% der Daten liegen, die obere und untere Begrenzung stellen jeweils die Grenzen des oberen und unteren Quartils dar. Getrennt werden die Quartile mittels dem Median. Die Antennen (Whiskers) zeigen jeweils das Minimum und das Maximum der gemessenen Daten auf.

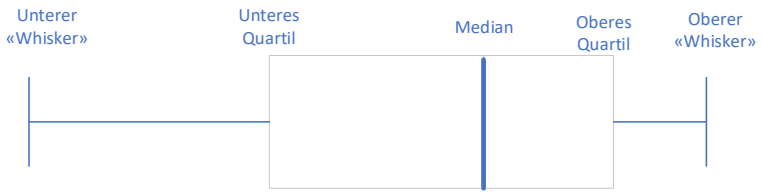


Abbildung 28: Aufbau eines Boxplots

Die Responses des Base-Models werden jeweils orange gezeichnet, die restlichen grau. Durch das Anwählen eines Boxplot werden per Tooltip Eckdaten der jeweiligen Response angezeigt. Die Boxplot-Ansicht kann durch das Mausrad vergrössert bzw. verkleinert werden. Wie auch in der Vergleichsansicht, können die Boxplots gefiltert werden. Dazu wird dasselbe Optionsmenu genutzt wie beim Modellvergleich. Die Boxplots werden durch die Selektion eines Experiment dynamisch hinzugefügt oder entfernt.

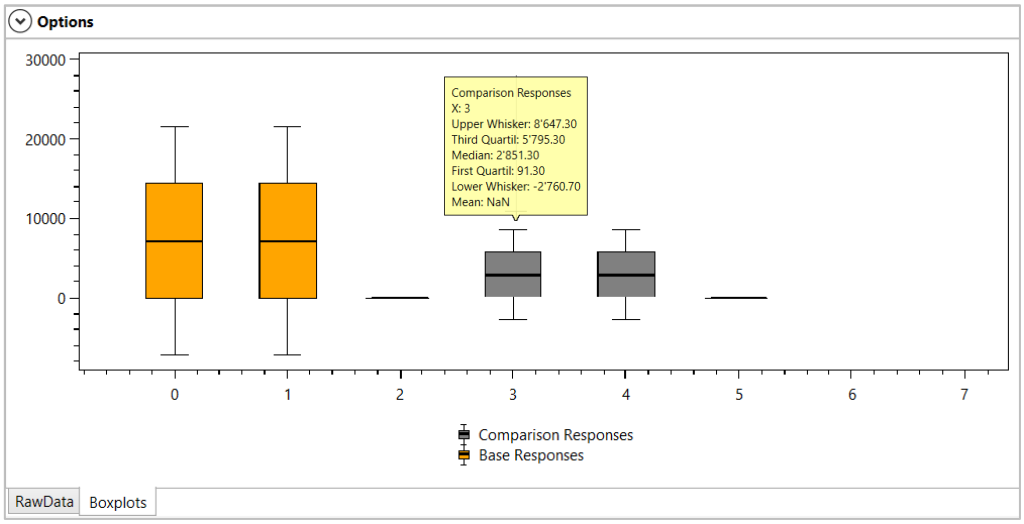


Abbildung 29: Visualisierung der Responses als Boxplots

8.4 Simio Community

Um Simio hat sich eine Community gebildet, welche als primärer Kommunikationskanal das von Simio bereitgestellte Forum nutzt. Um schon während des Entwicklungsprozesses Feedback einzuholen, wurde ca. in der Mitte des Projektes ein Prototyp auf diesem Forum bereitgestellt. Der Forumseintrag[5] beinhaltet eine textuelle Beschreibung und folgende Produkte:

- Installer
- ModelAnalyzer-Prototyp
- Kurzanleitung

Um Feedback einzuholen ist eine Online-Umfrage erstellt worden. Diese enthielt folgende Fragestellungen zu den Bereichen Usability, Funktionalität und Fehler/Bugs.

Usability

- How would you describe the overall design of the Plugin?
- Were the symbols/icons used easy to understand?
- How do you rate the navigation?
- Are the functions clearly defined by its symbols?
- What would you change about the visual appearance?
- Comments regarding the usability of the tool

Nutzen/Grundfunktionalität des Add-Ins

- Would you use this application in your simulation projects?
- What was the best function?
- Which function was the most frustrating?
- Were you able to clone a model?
- Were you able to compare two models?
- Which functionality is missing?
- Would this tool support you in an efficient way?
- What tasks did you do with the tool? Were you able to achieve your goal?
- Please rate the overall functionality of the tool
- Comments regarding the functionality of the tool

Bugs/Errors

- Did you experience any bug, unexpected or undefined behavior while using the tool?

Die Dauer der Umfrage war auf drei Wochen begrenzt. Der Grundgedanke war, dass eingehendes Feedback direkt umgesetzt werden kann. Leider ist kein Feedback eingegangen, was eine Auswertung hinfällig macht.

[5] Forumseintrag einsehbar unter: https://www.simio.com/forums/viewtopic.php?f=36&t=2856

9. Testing

9.1 Unittest

Im Rahmen der Testabdeckung sind Unittests geschrieben worden. Dabei wurde ein grosser Fokus auf die Korrektheit des Differenztests gelegt. Die Unittest sind jeweils nach Layer aufgeteilt und in separate Assemblies ausgelagert worden.

9.2 Testszenarien

Zur Überprüfung der einzelnen Funktionalitäten sind diverse Szenarien erstellt worden. Jedes Szenario deckt jeweils einen Funktionalitätszweig ab und zielt darauf ab, die Anwendung aus Benutzersicht zu testen. Während der Entwicklung wurden diese Szenarien mehrfach wiederholt durchgeführt. Dadurch konnten Fehler und das korrekte Zusammensiel aller Komponenten ermittelt werden.

Gesamtübersicht		
Szenario		**OK?**
1	Installation	x
2	Modelmanagement	x
3	Modellvergleich	x
4	Vergleichsergebnisse strukturieren und visualisieren	x
5	Standalone	x

10. Anwendungsbeispiel: Photobook

Zum Abschluss dieser Bachelorarbeit wird das Zusammenspiel von Simulationsspirale und ModelAnalyzer aufgezeigt. Dies wird anhand eines simplen Beispielprojekts aufgezeigt. Das Beispielprojekt verläuft nach dem entwickelten Modellierungsprozess. Die Modellvarianten innerhalb von Simio werden mit dem ModelAnalyzer geklont und verglichen. Klar im Vordergrund stehen die Anwendung des ModelAnalyzer und der Simulationsspirale als Vorgehensweise.

10.1 Problemstellung

Untenstehendes Wortmodell beschreibt die Problemstellung.

Photobook

Ein Startup-Unternehmen plant eine Fotobuchdruckerei. Dazu hat sie bereits Daten erhoben um gewisse Randbedingungen abzustecken. Nun möchte das Startup eine Simulation ihres Betriebs durchführen, um so Engpässe und Auslastung der einzelnen Teilprozesse und deren Maschinen zu ermitteln. Folgendes Wort-Modell ist dazu angefertigt worden.

Es werden im Mittel mit 500 Aufträgen pro Tag gerechnet. Ein Druckauftrag kann auf drei gleichen Maschinen bearbeitet werden und dauert im Mittel 7 Minuten. Dem Prozess sin die Seitenzahlen nach oben und nach unten begrenzt, daher wird die Dauer eines Auftrags zwischen 3 und maximal 15 Minuten dauern. Anschliessend gehen die Fotos in die Kommissionierung (Random.Triangular(9,10,11)). Diese beinhaltet entweder ein aufwändiges Binden oder ein leichteres, schnelleres Kleben. Ein grosser Teil der Aufträge wird gebunden. Danach werden die Fotobücher für den Versand fertig gemacht, was im Mittel nur eine Minute dauert. Die durchschnittliche Zeit zur Bearbeitung eines Jobs vom Auftragseingang, bis zur Fertigstellung zum Versand dauert ca. 4.5 Stunden.

Problem: Identifizieren von Engpässen, optimieren der Verarbeitung.

10.2 Erste Ebene

Die erste Ebene im Modellierungsprozess abstrahiert das Wortmodell. Zuerst werden die relevanten Daten im Zuge der isolierenden Abstraktion ermittelt. Anschliessend werden in der zweiten Phase die verschiedenen Prozesses analysiert und formalisiert. Das Ergebnis dieses Schritts ist ein formales Modell. Aus dem formalen Modell entstehet danach das programmierte Simio-Modell. Dieses wir mit entsprechenden Experimenten bestückt.

10.2.1 Isolierende Abstraktion

Während dieser Phase gilt es, möglichst genaue Daten zum Realsystem zu ermitteln.

Information	Wert
Aufträge pro Tag	Mittel: 500
Druckdauer	Mittel: 7 Minuten Minimum: 3 Minuten Maximum: 15 Minuten
Kommissionierung (Binden/Kleben)	Random.Triangular(9,10,11)
Überführung zum Versand	Mittel: 1 Minute
Auftragsdauer (über alle Schritte)	Durchschnitt: 4.5 Stunden

Weiter ist zu berücksichtigen, dass momentan drei Maschinen eingesetzt sind, welche Druckaufträge bearbeiten können.

10.2.2 Prozess-Analyse - Formales Modell

Der Gesamtprozess, welcher im Wort-Modell beschrieben ist, kann in verschiedene Teilschritte zerlegt werden. Folgende Teilprozesse können identifiziert werden:

- Auftragseingang
- Drucken
- Kommissionieren (Kleben/Binden)
- Vorbereiten zum Versand

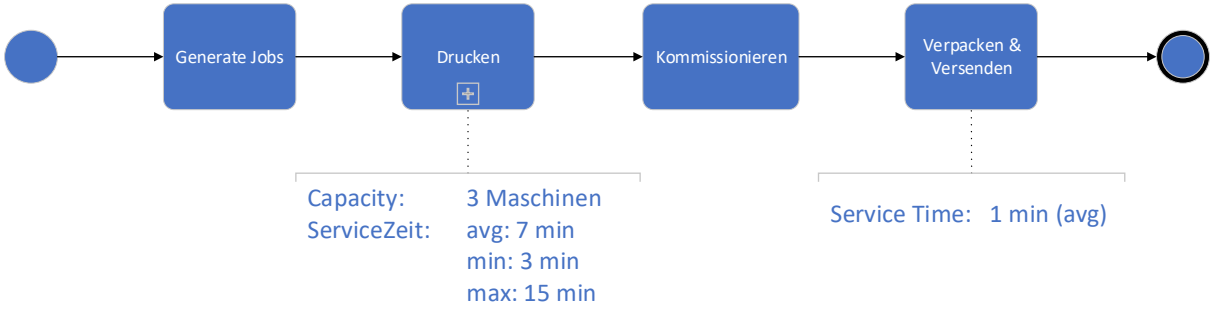


Abbildung 30: Formales Modell Photobook" in BPMN

Durch die gegebenen Eckdaten können bereits erste Faktoren errechnet werden. Ein relevanter Faktor stellt hier die Utilization der einzelnen Maschinen dar, welche über folgende Formel berechnet werden kann:

$$Ankunftsrate * Servicezeit = Utilization$$

$$\lambda * S = U$$

Teilschritt/Maschine	Wert [Utilization]
Gesamtprozess	$\lambda = \frac{500}{24\ h} = \frac{500}{1440\ min} = \mathbf{0.3472}\ {}^{1}/_{\boldsymbol{min}}$
Drucker	$\frac{\lambda S}{3} = \frac{0.3472\ {}^{1}/_{\min} * 8.3 min}{3} = \frac{2.892}{3} = \mathbf{0.961}$
Kommissionierung	$\lambda S = 0.3472\ {}^{1}/_{min} * 4\ \min = \mathbf{1.0416}$
Verpacken	$\lambda S = 0.3472\ {}^{1}/_{min} * 1\ \min = \mathbf{3.472}$

Bereits diese errechneten Werte geben Aufschluss darüber, dass ein Engpass entstehen wird. Dies ist dadurch identifizierbar, dass der Schritt Kommissionierung eine Utilization grösser als 1 aufweist. Dadurch wird auch der Schritt Verpacken einen Wert über 1 aufweisen.

10.2.3 Simio-Modell

Aus dem formalisierten Modell ist nun ein entsprechendes Simio-Modell entstanden. Dieses zeigt die einzelnen Teilschritte nun als konkrete Elemente auf. Die gesammelten Eckdaten sind nun als Parameter in die einzelnen Objekte eingesetzt.

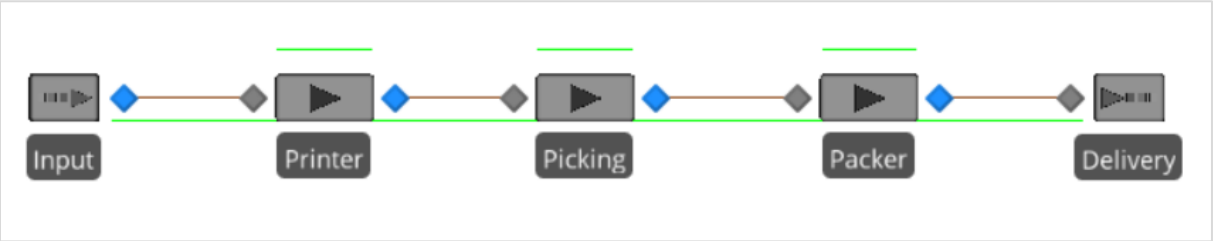


Abbildung 31: Simio-Modell der ersten Ebene

Um das Modell anschliessend zu validieren, wurde ein entsprechendes Experiment angelegt, welches die Utilization der einzelnen Arbeitsschritte misst.

10.2.4 Validierung

In dieser ersten Ebene wird das Modell mit den errechneten Daten und den Experimentresultaten verglichen. Daraus geht folgende Gegenüberstellung vor:

Teilschritt	Errechnete Werte	Experimentresultate
Drucken	0.9610	0.9344
Kommissionierung	1.0416	0.9954
Verpacken	0.3472	0.0990

Durch die Auswertung der Gegenüberstellung geht hervor, dass die Auslastung des Teilschritts Verpacken eine zu niedrige Auslastung aufweist. Dies bedeute, dass die Aufträge durch einen Engpass aufgestaut werden. Dies hat direkten Einfluss auf den Gesamtprozess, welcher nicht den gewünschten Durchsatz erreicht. Der Engpass kann der Kommissionierung zugeordnet werden. Zusätzlich zum

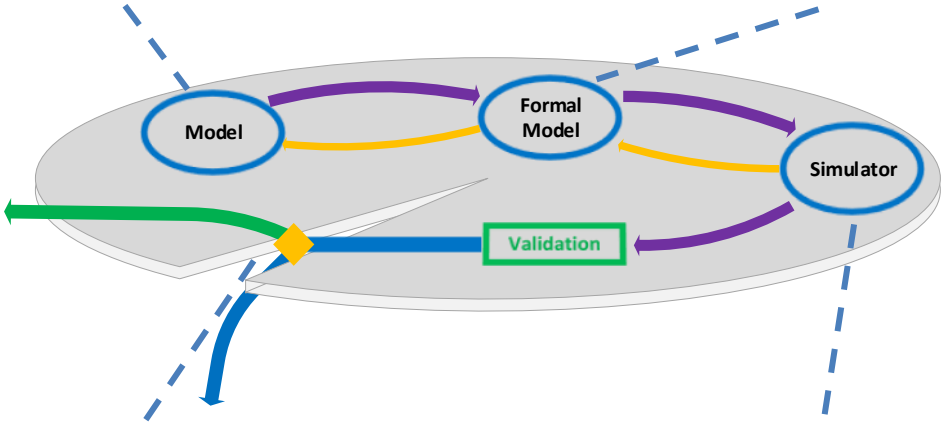
Engpass bildet der Schritt der Kommissionierung die Realität nicht hinreichend ab. Es wird keine Unterscheidung auf die Verarbeitungstypen Kleben und Binden gemacht.
Aufgrund dieser Validierung wird das Modell als nicht genügend an die Realität angenähert angesehen, es ist nicht valide. Die Konsequenz dieser Validierung ist das Einführen einer neuen Ebene.

10.3 Verlassen der ersten Ebene

Folgende Darstellung der aktuellen Ebene der Simulationsspirale zeigt die konkreten Tätigkeiten innerhalb der einzelnen Phasen kurz auf. Der letzte Punkt zeigt den Einsatz des ModelAnalyzers auf.

- Abstraktion des Wort-Modells
- Sammeln der Eckdaten
- Isolieren der relevanten Daten

- Analyse des Gesamtprozesses
- Unterteilung in verschiedene Subprozesse
- Formalisierung durch BPMN



- Validierung negativ ausgefallen
- **Klon erzeugen (ModelAnalyzer)**
- Neue Ebene einfügen

- Implementierung des formalen Modells
- Modellierung des Simio-Modells
- Ermitteln erster Simulationsergebnisse
- Engpässe sind erkannt worden

ModelAnalyzer

Durch diese erste, negativ ausgefallene Validierung kommt der ModelAnalyzer zum Einsatz. Durch ihn wird die bestehende Modellvariante nun geklont und bildet somit die Grundlage der nächsten Ebene des Modellierungsprozesses.

10.4 Zweite Ebene

Die Validierung auf der ersten Ebene ist negativ ausgefallen. Um den Modellierungsprozess weiterzuführen, wird nun eine weitere Ebene erzeugt. Die bestehende Modellvariante wird geklont und bildet die Grundlage der zweiten Ebene. Der Klon durchläuft nun innerhalb der zweiten Ebene die drei Simulationsphasen erneut.

10.4.1 Isolierende Abstraktion

Die Kommissionierung stellt einen kritischen Faktor dar. Aus dem Wortmodell kann ermittelt werden, dass es zwei verschiedene Arten der Endverarbeitung gibt, Kleben und Binden. Die beiden Teilschritte werden durch die Wörter aufwändiger (Binden) und schneller, einfacher (Kleben) quantifiziert.

Durch diese Aufteilung können durch das Treffen von Annahmen die Utilization-Werte für die beiden neuen Teilschritte ermittelt werden. Es wird angenommen, dass 80% aller Aufträge geklebt und lediglich 20% gebunden werden.

Teilschritt/Maschine	Wert [Utilization]
Binden	$0.2\,\lambda\,S = 0.2 * 0.3472\ {}^{1}/_{min} * 10\,\text{min} = \mathbf{0.694}$
Kleben	$0.8\,\lambda\,S = 0.8 * 0.3472\ {}^{1}/_{min} * 2\,\text{min} = \mathbf{0.5}$

10.4.2 Prozess Analyse

Aus der vorgehenden Phase geht hervor, dass die Kommissionierung in zwei Teilschritte unterschieden werden muss. Im formalen Modell bedeutet dies, dass der Teilschritt «Kommissionierung» nun in zwei separate Aufgaben zerlegt wird.

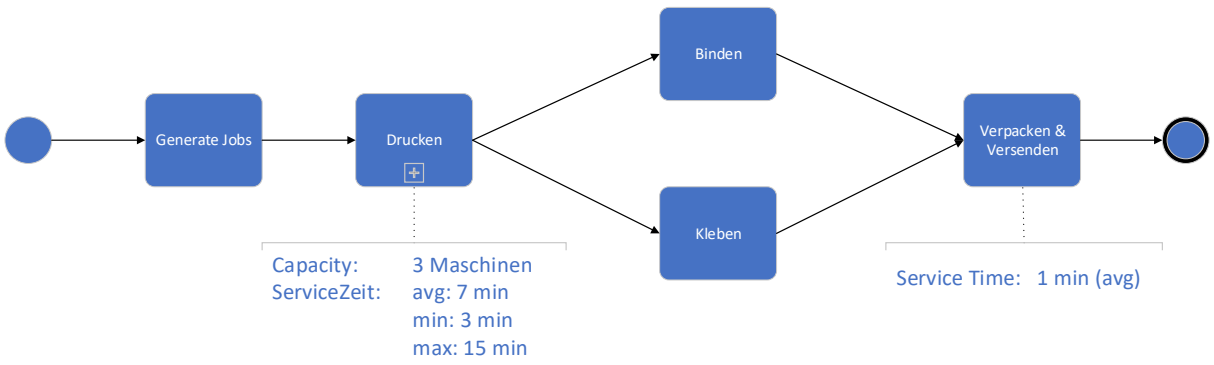


Abbildung 32: Formales Modell der zweiten Ebene

10.4.3 Simio-Modell

Entsprechend des neuen formalen Modells, ist auch das Simio-Modell entsprechende angepasst worden. Das Element «Picking» wurde durch die beiden Server «Binder» und «Gluer» ersetzt. Entsprechend der getroffenen Annahmen wurden ihre Parameter gesetzt.

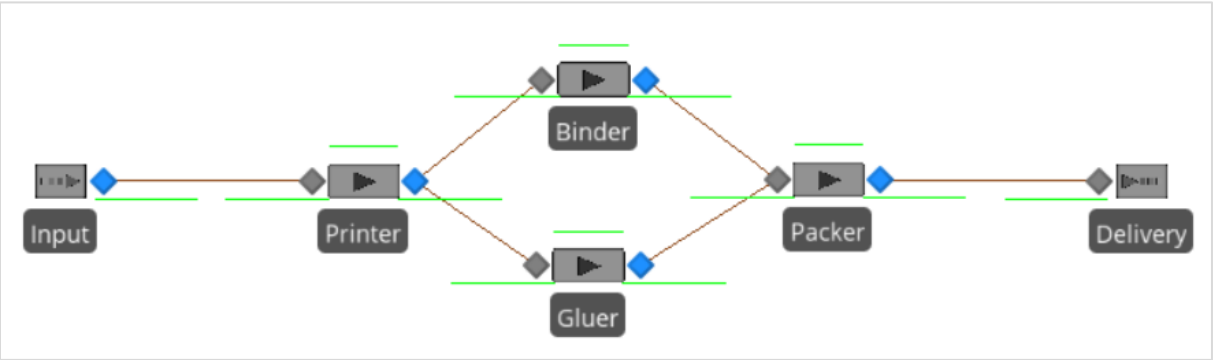


Abbildung 33: Simio-Modell auf der zweiten Ebene

Das bestehende Experiment wird um zwei Responses erweitert, welche die Utilization der beiden neuen Server abfangen.

10.4.4 Modellvergleich

Die beiden Modellvarianten können nun durch den ModelAnalyzer miteinander vergleichen werden. Durch diesen Schritt kann ermittelt werden, ob die generierten Werte voneinander abweichen und ob sie eine signifikante Abweichung aufweisen.

ModelAnalyzer

Beide Modellvarianten werden nun im ModelAnalyzer angezeigt. Durch das Auswählen der jeweiligen Experimente und das Setzen der Basis von der ersten Variante, werden die Modelle miteinander verglichen. Der Modellvergleich zeigt folgende Werte und Abweichungen auf:

Base	Model	PrinterUtil	PickerUtil	GluerUtil	BinderUtil	PackerUtil	TIS
•	Base_Level_One	93.4452	99.5433	-	-	9.9033	8.6885
	Clone_Level_Two	93.0596	-	53.5922	65.6512	33.29	0.3727

Die zweite Variante weist positive signifikante Abweichungen gegenüber der ersten auf. Falls bei diesem Vergleich keine oder negative Abweichungen entstanden wären, hätte diese zweite Variante verworfen oder erneut modifiziert werden können. Dadurch, dass nun positive Abweichungen aufgezeigt sind, wird diese Variante anschliessend validiert.

10.4.5 Validierung

Durch den vorgegangenen Modellvergleich sind signifikante Abweichungen gegenüber der ersten Modellvariante ermittelt worden. Nun gilt es, die generierten Werte gegenüber des Wort-Modells zu validieren. Dies erfolgt erneut durch eine Gegenüberstellung der Werte.

Teilschritt	Errechnete Werte	Experimentresultate
Drucken	**0.9610**	**0.9305**
Binden	**0.6940**	**0.6565**
Kleben	**0.5040**	**0.5359**
Verpacken	**0.3472**	**0.3329**

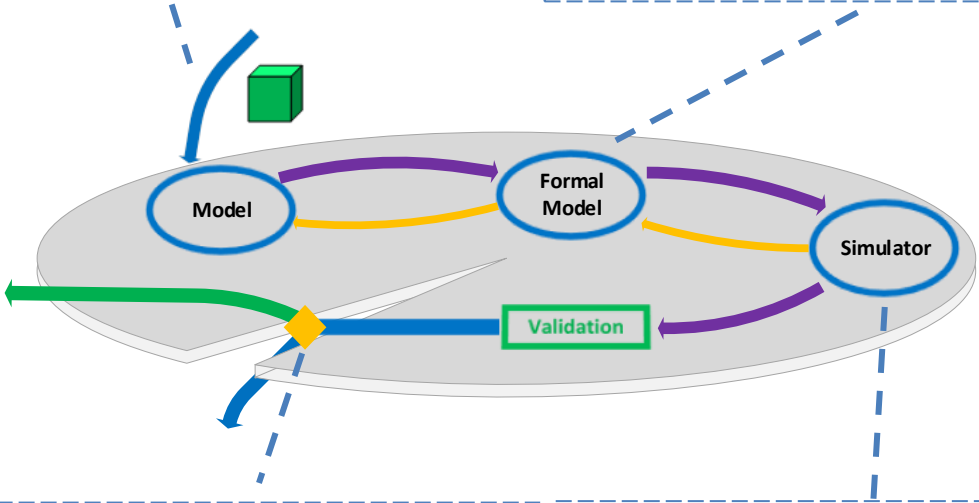
Die vorher kritischen Werte des Verpackens und des Gesamtprozesses haben sich nun den errechneten Werte angenähert. Die Werte liegen in einem annehmbaren Bereich was zu einer gültigen, positiv ausfallenden Validierung führt. Durch diese Entscheidung wird die zweite Ebene abgeschlossen.

10.5 Verlassen der zweiten Ebene

Durch die gültige Validierung der zweiten Modellvariante wird beim Verlassen dieser zweiten Ebene nun der alternative Weg gewählt. Dieser markiert den Abschluss der Spirale und somit des Modellierungsprozesses.

- Eintritt in neue Ebene
- Grundlage bildet Klon der ersten Ebene

- Abstrahierung und Isolation neuer Erkenntnisse
- Einfügen neuer Teilprozesse (Binden und Kleben)



- Validierung des neuen Modells
- Annäherung an Realität genügt
- Modell ist valide
- Modellierungsprozess ist beendet

- Umsetzen des formalen Modells
- Simulieren der neuen Modellvariante
- Vergleich der beiden Modelle **(Model-Analyzer)**
- **Signifikante Abweichungen ermittelt**
- Neues Modell weist bessere Werte auf

11. Schlussfolgerung und Ausblick

Die angestrebten Ziele und Anforderungen wurden erreicht. Durch den ModelAnalyzer werden ergänzende Funktionalitäten zur Verfügung gestellt. Zusätzlich ist ein API-Adapter entwickelt worden, welcher als Plattform für künftige Entwicklungen fungiert. Die Simulationsspirale stellt eine neue Vorgehensweise für Simulationsprojekte vor.

11.1 Implementation ModelAnalyzer

Die Funktionalität des ModelAnalyzers schliesst defizitäre Lücken von Simio. Es ist nun möglich, Modellvarianten aus bestehenden Simio-Modellen zu erzeugen und zu versionisieren. Dies ohne Verlust oder Einschränkungen in der Funktionsweise der Modelle. Durch die Vergleichskomponente können diese Varianten auf der Basis eines Differenztests mit unabhängigen Stichproben miteinander verglichen werden. Diese Modellvergleiche zeigen signifikante Abweichungen zwischen zwei oder mehreren Modellen auf. Verschiedene Vergleichsoptionen ermöglichen ein effizientes filtern und strukturieren der Vergleichsergebnisse. Die Weiterverarbeitung der Modelldaten ist durch die Exportfunktion gegeben. Die gesetzten Anforderungen an den ModelAnalyzer sind alle abgedeckt worden.

11.2 Implementation Adapter.SimioAPI

Als Grundlage zur Implementierung ist ein API-Adapter entwickelt worden. Dieser kapselt momentan Teile der SimioAPI und stellt diese als zentraler Service zur Verfügung. Dadurch kann völlig unabhängig und entkoppelt von der SimioAPI entwickelt werden. Diese Plattform kann leicht erweitert werden.

11.3 Simulationsspirale

Die während der Implementierung entwickelte Simulationsspirale stellt eine neues Modell zur Vorgehensweise in Simulationsprojekten vor. Durch den iterativen Aufbau in Form der Ebenen, werden wesentliche Phasen mehrmals durchlaufen. Jede Modellvariante durchläuft den Prozess der Validierung. Durch das Einfügen von weiteren Ebenen, verfeinern sich die einzelnen Modellvarianten fortlaufen, bis sie sich an den gewünschten Grad der Realität angenähert haben oder diese sogar vollständig abdecken.

Unterstützt wird dieses Modell durch den ModelAnalyzer. Im Zusammenspiel von Modell und Implementation wird eine neue Entscheidungsbasis für den Hypothesentest geschaffen.

11.4 Ausblick

Durch den Einsatz des ModelAnalyzers wird zeigen, ob die realisierte Funktionalität den Modellierungsprozess hinreichend unterstützt. Durch das durchgeführte Beispielprojekt sind jedoch schon viele angenehme Einsatzmöglichkeiten eingetroffen. Als denkbare Erweiterung des ModelAnalyzers ist eine Gewichtung von einzelnen Responses denkbar.

Durch die Entwicklung des Installer und des API-Adapters sind solide Grundlagen für zukünftige Entwicklungen von Simio-Add-Ins an der HSR gelegt worden. Durch ihre leichte Erweiterbarkeit und der einfachen Zugänglichkeit sind viele Hürden im Zusammenhang mit Simio und dessen API überwunden worden.

Die Simulationsspirale muss zu ihrer Verifikation und Tauglichkeit an verschiedenen Projekten erprobt werden. Diese sind von Vorteil umfangreicher wie das gezeigte Beispielprojekt. Der Grundstein ist gesetzt, Verfeinerungen und Optimierungspotential zeigen sich mit dem Einsatz und der Zeit.

12. Abbildungsverzeichnis

13. Quellenverzeichnis

[1] Prof. Dr. A. Rinkel, «SMS - System Modeling and Simulation, Lecture 3 Grundlagen Simulation».

[2] Prof. Dr. A. Rinkel, «SMS - System Modeling and Simulation, Lecture 7 - Verifikation und Validation».

[3] Prof. Dr. A. Rinkel, «SMS - System Modeling and Simulation, Lecture 4 - Modellbildung».

[4] Prof. Dr. A. Rinkel, «SMS - System Modeling and Simulation, Lecture 7 - Schätzen und testen zur Überprüfung der Simulationsergebnisse».

[5] SQLite Consortium, «SQLite,» 06 04 2018. [Online]. Available: https://sqlite.org/.

[6] Microsoft, «Implementing the Model-View-ViewModel Pattern,» 08 02 2010. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff798384(v=pandp.10).

[7] C. Peiris, «Hosting and Consuming WCF Services,» Microsoft, 2007. [Online]. Available: https://msdn.microsoft.com/en-us/library/bb332338.aspx.

[8] Microsoft, «Dependency Injection with Unity,» 25 05 2012. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff647202(v%3dpandp.10).

[9] «WIX Toolset,» 31 12 2017. [Online]. Available: http://wixtoolset.org/.

[10] Simio LLC, «Simio,» [Online]. Available: http://www.simio.com/.

[11] Simio LLC, «Simio Forum,» 05 10 2017. [Online]. Available: https://www.simio.com/forums/.

14. Glossar

Add-In
Erweiterung von Software ... 10
API
Application Programming Interface, Programmierschnittstelle ... 69
BPMN
Business Process Modelling Notation ... 9
Constraints
Einschränkung, Rahmenbedingung ... 21
CSV
Comma Separated Values ... 30
DLL
Dynamic Linked Library ... 39
GUID
Global Unique Identifier ... 31
Mean
Mittelwert ... 15, 16
MVVM
Model View ViewModel - Entwurfsmuster ... 28
NFR
Non-functional Requirement, Nicht-funktionale Anforderung ... 21
Response
Kapselt spezifischen Wert eines Elementes und Scenarioresultate ... 10
SAD
Software Architecture Document ... 4
Simio
Simulationssoftware ... 10
Use-Cases
Anwendungsfalls aus Sicht des Benutzers ... 24, 25, 26, 69, 70
WCF
Windows Communication Foundation ... 34
WPF
Windows Presentation Foundation ... 28