Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Fachstudie Nr. 194

Untersuchung der Interaktionsmethoden und vorausgesetzten Programmiererfahrung von Simulationswerkzeugen

Steven Großmann, Johannes Herter, Nicholas Rush

Studiengang:	Softwaretechnik
Prüfer/in:	Jun.-Prof. Niels Henze
Betreuer/in:	Dipl.-Inf. Miriam Greis Dipl.-Inf. Lars Lischke
Beginn am:	14. April 2014
Beendet am:	14. Oktober 2014
CR-Nummer:	I.6.m

Kurzfassung

Der SimTech Cluster der Universität in Stuttgart vereint viele Experten zum Thema Simulationen. Da diese Experten aus den unterschiedlichsten Fachrichtungen kommen, bzw. die unterschiedlichsten Dinge simulieren und modellieren, sind auch einige Mitglieder dabei, die wenig oder keine Programmiererfahrung aufweisen können. Auf Grund dieses Umstandes soll ein Simulationsprogramm entwickelt werden, dass möglichst wenig Programmiererfahrung voraussetzt. Im Vorfeld zur Entwicklung eines solchen Programms sollen, im Rahmen dieser Fachstudie, bereits im SimTech verwendete Programme untersucht werden. Dabei liegt das Hauptaugenmerk auf den Interaktionsmethoden und dem Grad der Programmiererfahrung. Mit Hilfe einer Onlineumfrage wurde eine Übersicht von verwendeten Programmen, die auch von den jeweiligen Teilnehmer der Umfrage bewertet wurden, gesammelt. Außerdem wurden Interviews mit einigen Mitarbeitern von SimTech geführt, die einen tieferen Einblick in die Verwendung der Programme liefern sollte. Basierend auf den Erkenntnissen der Umfrage und der Interviews wurde eine Empfehlung für die Entwicklung des Simulationswerkzeugs ausgesprochen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Seit ca. einem halben Jahrhundert werden Simulationen in den verschiedensten Fachgebieten genutzt [Gorb]. Während die Geschichte der Simulation beim Militär beginnt, findet sie heutzutage in der Medizin, der Wirtschaft, den Sozialwissenschaften, dem Maschinenbau und vielen anderen Feldern Anwendung [SB11]. Dabei wird in den meisten Fällen nicht die reine Simulation, sondern der ganze Simulationsprozess (in der Literatur „Modeling and Simulation“ oder M&S [SB11]) betrachtet. Der Simulationsprozess besteht aus mehreren Schritten, die von der Erstellung eines Modells, über die Simulation, bis hin zur Visualisierung und Analyse reichen [SB11]. Dabei wird ein Modell als Repräsentation der realen Welt beschrieben, das Abläufe oder Ereignisse widerspiegelt. Die Simulation selbst ermöglicht dann, das Verhalten des Modells unter gegebenen oder sich ändernden Bedingungen zu beobachten.

Diverse Simulationsprogramme wurden bereits für die verschiedensten Fachgebiete entwickelt [Gora]. Multifunktionale Simulationsprogramme sollen Simulationen von Systemen aus verschiedenen Gebieten ermöglichen. Dadurch muss lediglich die Bedienung eines Programms erlernt werden, was zusätzlichen Lernaufwand und weitere Kosten vermeidet. Multifunktionale Simulationsprogramme benötigen deshalb eine Modellierungssprache, mit der unterschiedliche Probleme sinnvoll dargestellt werden können. Besonders diagrammbasierte Modelle konnten sich für diese Aufgabe qualifizieren [Gora]. Allerdings sind diese meist durch eine feste Auswahl an Komponenten beschränkt. Komplexe Systeme benötigen jedoch Individualisierungsmöglichkeiten, um das System möglichst originalgetreu modellieren zu können. Mit Hilfe von speziellen Interaktionsmethoden soll auch Programmieranfängern ermöglicht werden, individuelle Funktionen zu implementieren [KP05]. Hierfür sind grafische Oberflächen oder Methoden wie Programmieren durch Demonstrieren (Programming-By-Demonstration (PBD)) [DCST00] besonders geeignet.

Auch die Universität Stuttgart hat sich der Forschung im Gebiet der Simulationen gewidmet und vereint dazu Experten aus unterschiedlichsten Feldern im Stuttgart Research Centre for Simulation Technology [1] (SRC SimTech). Zusätzlich ist der Exzellenzcluster Simulation Technology (SimTech), welcher eine Graduiertenschule und einen Studiengang beinhaltet, in das SRC SimTech integriert.

Da nicht alle Experten des SimTech Clusters auch ausgeprägte Programmiererfahrung haben, soll ein Simulationswerkzeug entwickelt werden. Dieses Werkzeug soll unter der Prämisse entwickelt werden, dass möglichst wenig Programmiererfahrung zur Benutzung nötig ist. Ziel dieser Arbeit ist es, eine Empfehlung für ein solches Simulationswerkzeug auszusprechen, wobei der Fokus auf den Interaktionsmöglichkeiten und der nötigen Programmiererfahrung liegt.

[1] http://www.simtech.uni-stuttgart.de

Um einen Überblick der aktuell verwendeten Simulationswerkzeuge im SimTech Cluster zu bekommen, wurde eine Umfrage erstellt. Diese Umfrage wurde mit Hilfe von einigen Mitarbeitern des SimTech Clusters verfeinert und dient nicht nur zur Bestandsaufnahme. Zusätzlich wurden noch weiter Informationen zu Interaktionsmöglichkeiten und nötiger Programmiererfahrung in diesen Programmen abgefragt. Die so ermittelten Werkzeuge wurden auf die nötige Programmiererfahrung und die entsprechenden Interaktionsmöglichkeiten untersucht. Anhand dieser Untersuchung wurde dann die Empfehlung ausgesprochen.

Diese Arbeitet gliedert sich in folgende Teile:

Kapitel 2 – Verwandte Arbeit: Definitionen und Erklärungen zum Thema Simulation und Modellierung.

Kapitel 3 – Umfrage: Aufbau der Umfrage und Präsentation der Ergebnisse, sowie Diskussion zu den Ergebnissen.

Kapitel 4 – Interviews: Zusammenfassung der abschließend geführten Interviews.

Kapitel 5 – Empfehlung: Anmerkungen und Empfehlung zur Umsetzung eines Simulationswerkzeugs.

Kapitel 6 – Zusammenfassung: Überblick über die Arbeit, die Ergebnisse und unsere Empfehlung.

2. Verwandte Arbeit

Durch die Verwendung von Simulationen in vielen verschiedenen Anwendungsgebieten, werden auch viele wissenschaftliche Arbeiten über Simulationen veröffentlicht. Das folgende Kapitel stellt eine Übersicht von Themenverwandten Arbeiten dar. Dabei liegt der Fokus auf grundlegenden Informationen zu Simulationen, Programme und Forschungen zum Thema intuitive Bedienung und Programme mit generische Anwendung. Zunächst werden Definitionen von wichtigen Begriffen wie Modellen, Simulationen und Systemen geklärt. Im Anschluss werden die einzelnen Schritte des Simulationsprozess genauer betrachtet und zusätzlich gängige Simulationsmethoden erläutert. Außerdem werden auch einige Vor- und Nachteile von Simulationen und bereits existierenden Simulationsprogrammen diskutiert. Des Weiteren werden multifunktionale Simulationsprogramme vorgestellt und Programmierterminologien sowie zugehörige Methoden erläutert

2.1. Definitionen

Modelle sind Abschätzungen bzw. Annäherungen an die reale Welt [SB11]. Dabei repräsentiert ein Modell ein Ereignis, Objekt oder komplexes System, das nicht notgedrungen existieren muss [SB11]. Auch die Repräsentation eines fiktiven Objekts wird als Modell gesehen. Dabei können Modelle in verschiedenen Abstraktionsebene entstehen. So stellt die einfache Zeichnung eines Autos ein relativ simples Modell dar, das wenig Einblicke in Abläufe und Funktionsweise eines Autos gibt. Eine Konstruktionszeichnung desselben Autos ist dabei ungleich Aufschluss- und Detailreicher und ermöglicht tiefere Einblicke in die Funktionsweise des Autos. Nicht unbedingt muss ein Modell das gesamte System erfassen. Es können auch kleine Teile des Systems modelliert und betrachtet werden. Modelle sind nicht auf grafische Repräsentationen beschränkt. Auch mathematische oder physikalische Formeln stellen Modelle dar.

Simulationen dienen zur Überprüfung und Validierung von Modellen und sollen das Verhalten eines Modells mit unterschiedlichen Startwerten, oder einfach über die Zeit hinweg zeigen. Einfach gesagt, ist eine Simulation die Rekonstruktion des Ablaufs eines beobachteten oder erfundenen Prozesses [SB11]. Zu beachten ist, dass eine Simulation eines grafischen Modells nur schwer durchzuführen ist. Mathematische oder physikalische Modelle eigenen sich besser zur Simulation. So kann zum Beispiel der Abrieb eines Reifens an einem Auto simuliert werden, wenn man vorher ein physikalisches Modell erstellt, das die unterschiedlichen Faktoren (Gewicht des Autos, Straßenbelag, etc.), die zum Abrieb führen, berücksichtigt. Die Ergebnisse einer solchen Simulation können verwendet werden, um zu zeigen, dass das Modell den Prozess der echten Welt geeignet repräsentiert (simulierter Reifenabrieb deckt sich mit den Beobachtungen am Auto). Wenn dies der Fall ist, kann man untersuchen, wie unterschiedliche Faktoren das Ergebnis der Simulation beeinflussen und daraus Rückschlüssel auf den Prozess schließen. Nutzen finden Simulationen vor allem bei Systemen, auf die man keinen Zugriff

hat (z.B. Sonnensystem), die zu nutzen zu gefährlich wären (z.B. Kernreaktoren), deren Veränderung inakzeptabel wäre (z.B. Ökosysteme), oder die einfach nicht existieren [SB11].

Wiederholt haben wir Systeme benutzt um andere Begriffe oder Prinzipien zu erklären. Dabei sehen wir Systeme als eine Sammlung kleinere Systeme oder auch Elementen, die durch Zusammenarbeit und Interaktion ein gemeinsames Ziel erreichen [SB11], [LKK91]. Wenn wir also ein Auto als System betrachten, fällt es uns nicht schwer kleiner Systeme oder Elemente in diesem System zu finden. Der Motor, die Elektronik und die Lenkung (und natürlich einige mehr) sind alles kleine Systeme, die auch ohne das jeweils andere funktionieren. Allerdings können sie nur zusammen die Aufgabe erfüllen, sich fortzubewegen. Elemente können also Maschinen und Bauteile sein, aber genauso können auch Software oder der Mensch als Element eines Systems vorkommen [SB11], [LKK91]. Dabei ist zu beachten, dass System meist in größere Systeme eingebettet werden können und es daher wichtig ist, den aktuellen Zusammenhang genau zu beachten.

2.2. Schritte im Simulationsprozess

Der Gesamte Simulationsprozess, in der Literatur häufig mit M&S („Modeling and Simulation“) abgekürzt, umfasst nicht nur das Durchführen der Simulation selbst. Zu Beginn des Prozess geht es darum das System, das simuliert werden soll, zu analysieren. Dazu gehört das Hinterfrage und Untersuchen von Teilsystemen, sowie die Prüfung von Wechselwirkungen mit anderen Systemen oder zwischen den Teilsystemen [SB11]. Dabei ist es wichtig möglichst genau und gewissenhaft vorzugehen, da anhand von den gesammelten Daten im nächsten Schritt des Prozesses ein Modell erstellt wird. Ziel bei der Erstellung des Modells ist es, eine möglichst genaue Repräsentation des Systems zu entwickeln. Das erstellte Modell wird im nächsten Schritt simuliert.Hier werden durch ändern gewisser Parameter im Modell, also quasi durchspielen unterschiedlicher Szenarien im System, verschiedene Abläufe und Ergebnisse der Simulation beobachtet und dokumentiert [SB11]. Um die Beobachtungen und Ergebnisse der Simulation verständlicher zu gestalten, werden im Schritt der Visualisierung die gesammelten Daten verarbeitet und in geeigneter Weise dargestellt. Dieser Schritt hilft uns beim Verständnis und der Interpretation der gesammelten Daten [SB11]. Zusätzlich kann mit Hilfe der Visualisierung auch der Ablauf der Simulation, z.B. mit 3D-Computer Grafiken, dargestellt werden [SB12]. Im letzten Schritt des Simulationsprozesses, der Analyse [SB12], werden die gesammelten Daten ausgewertet und interpretiert. Dabei können aufgestellte Thesen überprüft und validiert werden oder Rückschlüsse auf das simulierte System geschlossen werden [SB12]. Außerdem können Empfehlungen auf Basis der Simulation ausgesprochen und Rahmenbedingungen für das System vorgestellt werden [SB12].

2.3. Simulationsmethoden

Oft werden in der Literatur bestimmte Eigenschaften von Systemen, Modellen oder Simulationen hervorgehoben, die den Ablauf bestimmen. So kann ein System zum Beispiel auf Parametern beruhen, die sich langsam, mit dem Verlauf der Zeit, ändern, oder eben an einem bestimmten Zeitpunkt schlagartig verändert werden und so den Zustand des Systems ändern [SB11], [LKK91]. Zur einfacheren

Betrachtung grenzen wir in diesem Punkt nicht zwischen Systemen, Modellen und Simulationen ab, sondern betrachten den gesamten Simulationsprozess. Diese Zusammenfassung scheint uns sinnvoll, da z.B. ein dynamisches System auch meistens ein dynamisches Modell und eine dynamische Simulation nach sich zieht. Dadurch ergeben sich einige Simulationsmethoden, die wir in Gegensatzpaaren geordnet haben und im Folgenden erklären.

Eine statische Simulation ist dadurch definiert, dass entweder die Zeit keinerlei Rolle spielt, oder nur ein bestimmter Zeitpunkt betrachtet wird. Beispiel hierfür ist eine Monte-Carlo Simulation [LKK91]. Im Gegensatz zu statischen Simulationen stehen dynamische Simulationen. Bei diesen Simulationen spielt die Zeit eine entscheidende Rolle, da sich das betrachtete System mit der Zeit verändert bzw. entwickelt [LKK91].

Zusätzlich unterscheidet man zwischen deterministischen und stochastischen Simulationen. Eine deterministische Simulation basiert nicht auf probabilistischen Werte, beinhaltet also keine zufälligen Werte [LKK91]. Dabei ist zu beachten, dass eine solche Simulation durch die Eingabe von festen Werten auch zu einem festen Ergebnis determiniert. Stochastische Simulationen verwenden im Gegensatz dazu zufällige Variablen. Dabei können identische Startparameter zu unterschiedlichen Ergebnissen führen [LKK91]. Aus diesem Grund dürfen diese Ergebnisse nicht als tatsächliche Ergebnisse, sondern müssen als Abschätzung der Eigenschaft des Systems gesehen werden [LKK91].

Außerdem unterscheidet man auch in kontinuierliche und diskrete Simulationen. Bei beiden Arten geht es um die Veränderung von Variablen im Verlauf der Zeit. Bei diskreten Simulationen verändern sich Variablen zu bestimmten Zeitpunkten oder beim Auftreten von Ereignissen [SB11], [LKK91]. Diese Änderung tritt augenblicklich auf, ist also vergleichbar mit einem Schalter der umgelegt wird. Im Gegensatz dazu verändern sich bei kontinuierlichen Simulationen die Variablen fortlaufend mit der Zeit [LKK91].

2.4. Vor- und Nachteile von Modellierung und Simulation

Sokolowski und Banks [SB11] listen eine Reihe von Vor- und Nachteilen auf, die Modellierung und Simulation mit sich bringt. Dabei beziehen sie sich auf eine 1998 veröffentlichte Liste vom Institute of Industrial Engineers. Zunächst wollen wir einige Vorteile dieser Auflistung vorstellen:

- Die Fähigkeit sich richtig zu entscheiden, indem man jede mögliche Veränderung überprüft
- Komprimieren oder expandieren der Zeitlichen Abläufe, um dem Nutzer die Möglichkeit zur genaueren Untersuchung zu eröffnen
- Verständnis für Systeme, durch Erstellung und Untersuchung eines Szenarios
- Möglichkeiten erforschen, ohne das echte System zu stören
- Probleme diagnostizieren, indem die Interaktionen zwischen Variablen durschaut werden
- Visualisierung, um das System beobachten zu können
- Vorbereitung auf Änderungen, indem man mögliche Auswirkungen untersucht

- Vernünftige Investition, da eine Simulation billiger als die tatsächliche Änderung des Systems ist
- Kostengünstige Trainingshilfe

Durch die vielen Anwendungsmöglichkeiten ergeben sich auch viele Vorteile, die für die Verwendung von Modellierung und Simulation sprechen. Jedoch bringt es auch einige Nachteile mit sich, die von Sokolowski und Banks [SB11] auch weniger ausführlich beschrieben werden:

- Spezielles Training nötig, um Modelle zu erstellen
- Schwierige Interpretation von Ergebnissen, wenn diese auf Zufällen beruhen
- Zeitliche und finanzielle Kosten, da der Simulationsprozess aufwändig und kostenintensiv sein kann
- Falscher Einsatz von Simulationen, wenn z.B. ein analytischer Ansatz erfolgsversprechender wäre.

2.5. Multifunktionale Simulationsprogramme und Modellstrukturen

Seit den 1960er Jahren verstärkte sich das Interesse nach Programmen, welche komplexe Systeme simulieren können [Gorb].Über die Jahre entstanden deshalb viele verschiedenen Simulationsprogramme, die auf spezielle Probleme anwendbar sind. Allerdings müssen beispielsweise in Produktionsprozessen mehrere Systeme aus verschiedenen Gebieten simuliert werden [MI99]. Programme für spezielle Zwecke, benötigen meist eine spezielle Bedienung [MI99]. Aus diesem Umstand kristallisierte sich der Wunsch nach Simulationsprogrammen, die generisch in mehreren Gebieten, bzw. auf mehrere Problemstellungen anwendbar sind [Gorb]. Im Folgenden werden Modellstrukturen und Programme vorgestellt, die sich mit dieser Problematik beschäftigen. Dabei liegt der Fokus auf Modellstrukturen und Programmen, welche intuitiv und mit wenig bis keiner Programmiererfahrung bedient werden können. Zusätzlich werden die Vor- und Nachteile von generischen Simulationsprogrammen erläutert und diskutiert.

2.5.1. Gestaltung eines generischen Simulationsprogramms

Bereits in den 1960er Jahren erkannten Firmen wie IBM dass es nicht effizient ist, für jedes Fachgebiet ein spezielles Simulationsprogramm zu entwickeln [Gora]. Deshalb ergab sich der Bedarf an Simulationsprogrammen, die verschiedene Probleme lösen können [Gora]. Generische Simulationsprogramme müssen verschiedensten Anforderungen gerecht werden, da der Fokus dieser Programme nicht in einem speziellen Forschungsgebiet liegt. Aus diesem Grund muss eine Art der Modellierung gefunden werden, die ein breites Spektrum abdeckt und sich sinnvoll als Simulation ausführen lässt. Um einen Simulationsprozess zu initiieren müssen hauptsächlich zwei Schritte eingeleitet werden [Gorb]. Zuerst muss ein Modell für das System konstruiert werden, gefolgt von der Erstellung eines Programms welches die Logik und Aktionen des Modells produziert [Gorb]. Das lässt darauf schließen, dass der Grundstein für ein erfolgreiches, generisches Simulationsprogramm eine wohl definierte formelle

Sprache ist. Mit einem derartigen Programm kann alles simuliert werden, das die gegebene Sprache erfüllt. Die Herausforderung liegt darin, eine Sprache zu finden, die ein System möglichst detailliert beschreiben kann und dennoch generisch verwendbar ist.

Im Rahmen einer Untersuchung von fortgeschrittenen Schaltsystemen sollte ein Werkzeug entwickelt werden, um diese Systeme zu simulieren. Ein Projekt welches von J. P. Runyon geleitet wurde [Gora], verwendete für diese Aufgabe ein Simulationsprogramm das auf Sequenzdiagrammen basiert. Damit konnten Schaltknoten dargestellt werden, welche gerichtete Operationen initiierten. Dieses Programm wurde mit Verbesserungen erweitert, sodass mehr Details in die Simulation integriert werden konnten. So wurden die Knoten der Sequenzdiagramme beispielsweise um Zeitbeschränkungen oder spezielle Marker erweitert. In den Folgejahren wurde das Programm erfolgreich auf weitere Systeme angewandt. Neben Schaltsystemen wurden damit auch Verkehrsflusssimulationen ausgeführt. Die Verkehrsflusssimulationen wurden durch Bewegungen von Kunden in einem Supermarkt demonstriert. Mit diesem Beispiel sollte die Einfachheit und Generik des Simulationsprogramms bewiesen werden [Gora]. Für speziellere oder detailliertere Systeme mussten allerdings Anpassungen, bzw. Erweiterungen zu dem Programm hinzugefügt werden [Gora]. Bedauerlicherweise konnten die Sequenzdiagramme nicht für alle Systeme adaptiert werden. [Gora] Diesen Umstand bemerkte auch G. Gordon als er einem Projekt von Dr. D. V. Newton beitrat, indem er nach einer geeigneten Simulationsmethode für generische Systeme suchen sollte. Für große Entwicklungsprojekte konnte ein Simulationsprogramm mit einer Sequenzdiagrammnotation nicht sinnvoll skaliert werden [Gora]. Inspiriert von den Vorteilen der Sequenzdiagramme entwickelte Gordon eine Blockdiagrammnotation und ein Simulationsprogramm, zu Beginn unter dem Namen Gordon Simulator bekannt, das diese Notation verwendet [Gora].

Die Blockdiagrammnotation des Gordon Simulator

Blockdiagramme bestehen aus Blöcken, die mit Linienverbindungen kombiniert werden. Blöcke erfüllen bestimmte Funktionen in dem zu simulierenden System, während die Verbindungslinien einen Datenfluss repräsentieren. Die Durchführung einer Simulation geschieht durch die Erstellung von Transaktionen. Diese werden in einer chronologisch korrekten Reihenfolge und mit Rücksicht auf deren Prioritäten, durch die Blöcke weitergereicht [Gora]. Jede Bewegung, hier als Zustandswechsel anzusehen, bedeutet ein einzelnes Ereignis und geschieht zu einem bestimmten Zeitpunkt. Aus diesem Grund verwendet der Gordon Simulator eine Zeitschaltung, welche allerdings nur von Ereignis zu Ereignis schaltet. Das heißt, dass das System nicht in Echtzeit simuliert wird, sondern lediglich die Ereigniskette abarbeitet. Demzufolge haben Blöcke eine bestimmte Blockzeit, welche indiziert wie lange die Aktion des Blocks ausgeführt wird. Dadurch wird die Simulationsdurchführung beschleunigt und es werden lange Pausen ohne die Ausführung von Systemfunktionen vermieden [Gora]. Um einen korrekten Datenfluss im System zu gewährleisten, muss jeder Block mindestens einen Nachfolger besitzen. Durch einen bestimmten ”Selection Factor” wird entschieden, welcher Nachfolger gewählt wird [Gorb]. Teilweise benötigen Systeme eine Operationsausführung, während eine Transaktion durchgeführt wird. Das kann mit der Hilfe von Ausrüstungskomponenten implementiert werden. Die Komponenten können mit Transaktionen verknüpft werden und während der Transaktion ausgeführt werden. Transaktionen die temporäre Entitäten eines Systems repräsentieren, benötigen Attribute. Deshalb können Transaktionen Parameter übergeben werden. Beispielsweise halten Warenlager eine

bestimmte Anzahl an Waren, welche durch diese Attribute ausgedrückt werden können. Dadurch werden die Blockausführungszeiten während der Simulation beeinflusst.

In dem Programm von Gordon wird ein System durch verschiedene Block-Klassen definiert. Transaktionen werden mit Entstehungs- und Generierungsblöcken erstellt. Damit wird die Durchführung der Simulation eingeleitet. Entstehungs- und Generierungsblöcke können Transaktionen zur Laufzeit der Simulation generieren, manipulieren und entfernen [Gorb]. Weitere Block-Klassen bieten die Möglichkeit, Transaktionen mit Markierungen zu versehen. Die markierten Transaktionen können im weiteren Verlauf der Simulation an Fortschreitblöcken, Entscheidungen für die Nachfolgerblöcke treffen [Gorb]. So entscheiden beispielsweise an Transferblöcken die Markierungen der Transaktionen, über den weiteren Verlauf der Simulation. Des Weiteren gibt es Block-Klassen die Transaktionsausrüstungen kontrollieren, Block Speicher blockieren können und Statistiken sammeln. Um Ereignisse der Simulation interpretieren zu können, ist besonders letzterer Blocktyp wertvoll.

Das Allzweck-Simulationssystem - General Purpose Simulation System (GPSS)

Vorteilhaft für das auf Blockdiagrammen basierende Simulationsprogramm ist, dass Komponenten einfach ersetzt werden können [Gorb]. Während der Zeit der Entwicklung des GPSS, wollten Ingenieure und Analytiker es möglichst vermeiden selber programmieren zu müssen. Daraus resultierte die gute Organisation und Dokumentation, als wichtiger Faktor des GPSS. Des Weiteren geschieht die Erstellung einer Simulation ohne Programmierpraktiken und Programmierterminologie. Somit können auch Programmieranfänger das GPSS verwenden. Die Blockdiagrammsprache vermittelt somit die Illusion, dass der Anwender das System nicht programmiert, sondern es beschreibt [Gora]. Durch die Beachtung von Designregeln für Benutzerinteraktion, gewann das Programm an Mehrwert für unerfahrene Nutzer. So wurden Programmabbrüche bei Fehlimplementierungen, mit Informationen über den Systemstatus und Modellstatus in das Programm eingebaut. Allerdings sorgen bereits die hohen Strukturen des Modellprinzips dafür, Ausführungsfehler vorherzusehen und zu beseitigen. Es wurden auch Mechanismen und Automatismen in das Programm eingebaut, um dem Benutzer unnötig viel Interaktion zu ersparen. Wenn blockierte Transaktionen beispielsweise wieder als verfügbar gesetzt werden, startet der Algorithmus automatisch von vorne.

Das Programm wurde im Verlauf der 60er Jahre in diversen Anwendungsgebieten, wie an der Börse, im Städteverkehr, in Computerzentralen etc. verwendet und konnte sich in diesen Bereichen als sinnvoll beweisen [Gorb]. Jedoch führt der generische Ansatz des Simulationsprogramms zu Kompromissen, da es nicht immer den Anforderungen der Benutzer entspricht [Gorb]. Die Ausführungsgeschwindigkeit der Simulationen mit dem GPSS wurden ebenfalls kritisiert, da sie sehr langsam im Vergleich zu späteren Konkurrenzprogrammen ist.

2.5.2. Multifunktionale Simulationsplattformen und Pakete

Der Gordon Simulator bietet durch die Modellierung mit Blockdiagrammen einen generischen Ansatz Systeme zu simulieren[Gorb]. Allerdings sind beim GPSS von Gordon die Möglichkeiten, durch die vorgegebenen Blöcke begrenzt. Eine Funktion des Simulationsprogramms wurde in den öffentlichen Dokumentationen des Programms jedoch nicht erwähnt, nämlich die Integration von Hilfe-Blöcken

[Gora]. Gordon verwendete diese Funktionen lediglich zum Debuggen. Mit diesen Blöcken konnte selbst geschriebener Assembler Code in das Programm eingefügt werden, wodurch die Möglichkeit gegeben wurde eigene Block-Klassen zu definieren. Zwar können Anwender ohne Programmiererfahrung keine Blöcke selbständig implementieren, allerdings könnten programmiererfahrene Spezialisten diese Programme für andere Nutzer einrichten. Durch die daraus resultierende hohe Erweiterbarkeit des Simulators, gewinnt das Programm erheblich an Mehrwert für dessen Benutzer.

Compiler-Compiler für visuelle Sprachen

Programme wie die Software "Compiler-Compiler for Visual Languages" (CoCoViLa) [VK11] beinhalten diverse Möglichkeiten der Individualisierung. CoCoViLa ist eine Simulationsplattform die visuelle und modellbasierte Softwareentwicklung anbietet. Hierfür verwendet CoCoViLa eine strukturierte Darstellung von Programmen. Mit dieser Darstellung können deklarierte Spezifikationen von Simulationen in ausführbaren Programmcode umgewandelt werden. Für die Modellierung eines Systems, wird in CoCoViLa ein ähnliches Konzept wie im GPSS verwendet. Komponenten sind in diesem Programm das Pendant zu Blöcken im GPSS. Sie repräsentieren Java Klassen, mit zusätzlichen Annotationen die zur Darstellung der Komponenten dienen. Mit Hilfe der visuellen Werkzeuge der Plattform, können Spezifikationen ohne selbstgeschriebenen Programmcode erstellt werden. Anschließend werden die Spezifikationen vollkommen automatisch von CoCoViLa ausgeführt [VK11]. Während der Entwicklung der Plattform wurde sehr viel Wert auf Flexibilität gelegt, wodurch CoCoViLa multifunktional anwendbar ist [VK11]. Einzelne Komponenten lassen sich ohne Probleme mit bereits existierenden oder neu entwickelten Komponenten austauschen. Threads werden durch sogenannte Dämonen spezifiziert. Benutzerinteraktionen können durch Dämonen während Problembeschreibungen und während Simulationsphasen durchgeführt werden [VK11]. Simulationen können als fortschreitende Zeitsimulation oder als diskrete Ereignissimulation ausgeführt werden, was CoCoViLa flexibel einsetzbar gestaltet [VK11]. Mit der zusätzlichen Eigenschaft lässt sich das Verhalten eines Systems über einen Zeitraum betrachten, wodurch Echtzeitsimulation untersucht werden können.

Da es sich bei CoCoViLa um eine Plattform handelt, besteht die Möglichkeit verschiedene Designprinzipien zur Simulation anzuwenden [VK11]. Eines der Prinzipien beruht beispielsweise auf der automatischen Programmausführung von Spezifikationen. Dadurch wird ausführbarer Quellcode generiert. Da eine Spezifikation hier jedoch eine eigene Art von Quellcode ist, betrachten wir ein anderes Designprinzip. Personen ohne Programmiererfahrung können mit dem Designprinzip der modellbasierten Softwareentwicklung, ausführbare Modelle für Simulationen erstellen [VK11]. Mit Hilfe der visuellen Editoren der Plattform, können Simulationsdetails benutzerfreundlich editiert werden. Der Klasseneditor ermöglicht es, visuelle Aspekte der Komponenten mit Zeichnungen oder Bitmapimporten zu definieren [VK11]. Das trägt zu einer besseren Veranschaulichung der Komponenten und damit des kompletten Systems bei. Der Schema-Editor ist für mehrere Zwecke verwendbar. Damit können Simulationspakete, die im Klasseneditor erstellt wurden, importiert werden. Die Benutzeroberfläche wird individuell an die Paketbeschreibung anpasst. In diesem Editor werden auch die Simulationsprobleme visuell erstellt. Sie können dann in anderen Schemas als Komponenten in einer höheren Hierarchie integriert werden [VK11]. Zusätzlich bietet diese Oberfläche noch Optionen zum Debuggen an. Des Weiteren können Ergebnisse einer Simulation in einem neuen Fenster oder in

der Konsolenausgabe dargestellt werden. Es gibt noch weitere Teile der Plattform, wie zum Beispiel den Planer oder die Werkzeugbox. Mit dem Planer werden deklarierte Spezifikationen in ausführbaren Programmcode übersetzt. Dafür muss jedoch eine Spezifikation in der definierten Sprache der Plattform geschrieben werden [VK11]. Das ist nicht mit einer rein visuellen Oberfläche und ohne Programmierterminologie möglich.

20-SIM

Mit dem Modellierungs- und Simulationspaket 20-SIM, kann das dynamische Verhalten von Ingenieurssystemen modelliert und simuliert werden [Bro99]. Die Modelle in dieser Software, werden als hierarchisch strukturierte Bond-Graphen und Blockdiagramme erstellt. Zusätzlich lassen sich Gleichungen einfügen, um eigene Submodelle oder Gleichungsmodelle zu erstellen. Dadurch wird dem Benutzer mehr Freiheit in der Modellierung gewährt [Bro99]. Abgesehen von dem Gleichungseditor, können Modelle komplett mit einer grafischen Oberfläche und ohne Programmierpraktiken erstellt werden [Bro99]. Somit können die, im entsprechenden Kapitel beschriebenen, Blockdiagramme bequem und ohne Programmiererfahrung erstellt werden. Das gleiche gilt für die Bond-Graphen, welche einen essentiellen Teil des Pakets bilden. Durch diese Graphen können existierende Submodelle komfortabel wiederverwendet und erweitert werden [Bro99]. Im Vergleich zur Simulationsplattform CoCoViLa sind leider nicht die gleichen Individualisierungsmöglichkeiten gegeben. Jedoch ist 20-SIM etwas leichtgewichtiger und somit weniger kompliziert in der Bedienung. Durch den objektorientierten Ansatz von Bond-Graphen entstehen mehr Anwendungsmöglichkeiten, welche von Blockdiagrammen allein nicht kompensiert werden können. Besonders dann, wenn es sich bei der Modellierung und Simulation um physikalische Eingabe-/Ausgabesysteme handelt [Bro99].

2.5.3. Klassifizierung von Simulationswerkzeugen und Modellierungsprinzipien

Die bisher beschriebenen Simulationsprogramme, -Plattformen und -Pakete, sind nur ein kleiner Teil dessen, was bisher entwickelt wurde und auf dem Markt verfügbar ist [MI99]. Dies sollte lediglich veranschaulichen, wie Programme den Modellierungs- und Simulationsschritt mit verschiedenen Methoden und Techniken durchführen. Simulationswerkzeuge haben bestimmte Charakteristiken, die das Werkzeug klassifizieren. Beispielsweise definieren der Anwendungsumfang, das Modellparadigma und die Flexibilität, zu welchem Typ das Simulationsprogramm zugeteilt wird [MI99]. Die Wahl der Simulationsstrategie ist ein weiteres, signifikantes Merkmal [MI99]. Dabei gibt es hauptsächlich zwei Strategien, nämlich die Prozessinteraktion und die Aktivitätsabtastung. Im Prinzip können mit allen Strategien jegliche Problemstellungen gelöst werden [MI99], dennoch hilft die korrekte Wahl einer Strategie, die Modellierung und Simulation des Problems zu vereinfachen. Beispielsweise ist die Prozessinteraktions-Strategie besonders für Fertigungsanwendungen geeignet, in denen Materialien in ein System geliefert werden und einen eher statischen Prozessverlauf haben [MI99]. Hingegen ist die Aktivitätsabtastung-Strategie vorzugsweise für Systeme mit hoher Interaktion von Ressourcen, mit stark variierenden Zuständen, geeignet [MI99]. Die Möglichkeit mehrere Strategien zu verbinden besteht natürlich ebenfalls und wird des öfteren in Simulationswerkzeugen angewendet. Ein Beispiel ist hier der Drei-Phasen-Ansatz, bestehend aus Prozessinteraktion, Aktivitätsabtastung und Ereignisdisposition. Neben der Simulationsstrategie ist für generische Simulationswerkzeuge ebenfalls sehr

wichtig, welcher Modellierungsansatz verwendet wird. Dabei hat sich in vielen Dokumentationen von Simulationsprogrammen, mit Anwendungsmöglichkeiten ohne Programmierkenntnisse bewiesen, dass Modelle die auf visuellen Diagrammen basieren bevorzugt werden [Bro99] [Gorb] [VK11] [Gora] [MI99]. Hierbei müssen nämlich keine Programmierterminologien angewendet werden und das System kann praktisch mit Bauelementen zusammengesetzt werden [Gora]. Neben den bereits erwähnten Blockdiagrammen, Bond-Graphen und Sequenzdiagrammen [Gorb] [VK11] werden häufig auch Aktivitätsdiagramme oder Petrinetze verwendet [MI99], da sie eine ähnliche Struktur aufweisen. Zu beachten ist allerdings, dass auch nicht diagrammbasierte, sondern beispielsweise strukturbasierte Modelle, wie XML Code [TLR], für programmierunerfahrene Benutzer von Interesse sein können.

2.6. Programmierterminiologie und Methoden

Trotz der großen Auswahl an existierenden Simulationsprogrammen, konnte in unserer erstellten Umfrage (Kapitel **??**) festgestellt werden, dass viele Wissenschaftler ihre Programme selber entwickeln und in Simulationen verwenden. Das liegt unter anderem daran, dass diverse Programme nicht frei verfügbar sind oder dass für spezielle Probleme keine passenden Programme existieren. Für Personen ohne Programmierkenntnisse, stellt das ein Hinderniss zur Erstellung und Durchführung von Simulationen dar [KP05]. Neben dem Erlernen der strukturierten Herangehensweise an Problemstellungen, muss sich ein Programmieranfänger mit einer speziellen Syntax auseinandersetzen, die sich von einer natürlichen Sprache stark unterscheiden kann [KP05]. Diese Hürde wollen Forscher beseitigen, um das Programmieren einer breiteren Masse zugänglich zu machen. Kapitel 2.6.1 konzentriert sich auf das Erlernen von Programmiertechniken, unter der Voraussetzung dass der Programmieranfänger bereits Erfahrung mit strukturierten Herangehensweisen an Problemstellungen hat. Das bedeutet, dass die zum Programmieren benötigten Mechaniken und Konzepte [Sol86] vorhanden sind und lediglich die Kommunikation mit dem Computer eine Barriere darstellt.

2.6.1. Zugänglichkeit zum Erlernen neuer Programmiersprachen

Benutzer von Computersystemen kommunizieren in der Regel nur über Programme mit dem Computer, die sie nicht selber geschrieben haben [DCST00]. Das bedeutet, dass die wenigsten Computerbenutzer das Potenzial ihres Gerät ausnutzen können. Forscher erkannten diesen Mangel und arbeiteten seit den frühen 1960er Jahren daran, das Programmieren der breiten Bevölkerung zugänglicher zu machen [KP05]. Unglücklicherweise ist es den Forschern lange Zeit nicht gelungen, dieses Ziel zu erreichen [DCST00]. Der Grund dafür liegt allerdings nicht an mangelndem Interesse, sich eine Programmiersprache anzueignen, sondern mehr an der Barriere, welche die Zugänglichkeit zu einer neuen Sprache darstellt [DCST00]. Forscher haben über einen Zeitraum von ca. 30 Jahren diverse generische Sprachen an Programmieranfängern ausprobiert, dies allerdings ohne Erfolg [DCST00]. Somit kamen die Wissenschaftler zu dem Entschluss, dass die Programmiersprache selbst das Problem darstellt [DCST00]. Oft wirkt die Syntax einer Programmiersprache willkürlich und Anfänger können sich wenig unter den Abkürzungen und zusammengesetzten Befehlen vorstellen [KP05]. Deshalb versuchten die Sprachforscher Lernsysteme zu entwickeln, die Personen langsam an die Syntax und Semantik von Programmiersprachen heranführen. Erst nachdem die lernenden Personen Erfahrung

mit den Lernsystemen gesammelt hatten, wurden sie an generische, kommerzielle Programmiersprachen wie C++, Java, etc. herangeführt [KP05]. Dabei wird bei Lernsystemen darauf geachtet, dass sie eine gewisse Ähnlichkeit mit generischen Programmiersprachen haben, damit der spätere Wechsel der Studenten einfach zu bewältigen ist, ohne wieder eine komplett neue Herangehensweise erlernen zu müssen [KP05]. Des Weiteren haben Programmieranfänger oft Schwierigkeiten damit, ihre Absichten in korrekter Form als Text in einen Editor zu schreiben, wie es bei den meisten generischen Programmiersprachen üblich ist [KP05]. Deshalb gibt es zwei Ansätze, die ein Lernsystem verwenden kann, um Programmieranfängern den Start zu erleichtern. Entweder muss die Programmiersprache soweit verbessert werden, dass ein Anfänger diese Sprache leicht erlernen kann, oder es müssen Alternativen entwickelt werden, wie die Personen ihre Instruktionen in den Computer eingeben können [KP05].

Vereinfachen der Sprache: Generische Programmiersprachen beinhalten syntaktische Merkmale wie zum Beispiel die geschweiften Klammern in Java, oder die Labels in Fortran, welche für Anfänger schwierig zu verstehen sind, da sie keine offensichtliche Bedeutung zu haben scheinen [KP05]. Mit der Sprache Basic sollte dem entgegen gewirkt werden, da hier viele englische Begriffe eingebaut wurden, um dem Programmierer mehr Parallelität zu einer natürlichen Sprache zu gewähren [KP05]. Kritiker bemängelten, dass dadurch mehr Rechenaufwand beim Übersetzen der Sprache in Maschinencode verursacht wird. Doch da die Sprache die Programmierer nur an das Programmieren heranführen sollte, wurde das in Kauf genommen [KP05]. Eine weitere Lernprogrammiersprache ist Blue, welche bestimmte Restriktionen hat um das Programmieren für Anfänger zugänglicher zu gestalten. So gibt es in Blue die Beschränkung, dass alles nur auf eine bestimmte Weise programmiert werden kann, oder dass die Sprache sehr lesbar gestaltet ist, damit Studenten durch das betrachten von Beispielen verstehen um was es in dem Programm geht [KP05]. Besonders interessant bei Blue ist, dass es eine vollkommen objektorientierte Sprache ist und über Datei-Klassenstrukturen verfügt, sowie über einen Garbagecollector. Ebenfalls objektorientiert und zusätzlich nah an der Java-Syntax, da auch von Java abgeleitet, ist Junior Java (JJ). Der Code aus JJ kann als normaler Java Code exportiert werden, wodurch die Lücke zwischen Lernsprache und generischer Sprache klein gehalten wurde, um den späteren Wechsel zu erleichtern [KP05]. Das ist ein wichtiger Punkt, da die Balance zwischen Lernsprache und generischer Sprache sinnvoll sein muss, um einerseits den späteren Wechsel zu einer generischen Sprache zu vereinfachen. Andererseits muss eine möglichst einfache Terminologie verwenden werden, damit die Programmieranfänger einen leichten Einstieg haben und sich auf die Programmiermechanik konzentrieren können und nicht auf die Sprachsyntax.

Alternativen zur Computerinteraktion: Programmiersprachen unterliegen gewissen Beschränkungen, wie eine bestimmte Reihenfolge in der die Befehlen eingegeben werden müssen oder die korrekte Verwendung von Klammern. Mit diesen Eigenschaften kommen nicht alle Programmieranfänger zurecht [KP05]. Aus diesem Grund wurden Lernsysteme entwickelt, die eine andere Befehlseingabemethode verwenden, als über den herkömmlichen Texteditor. Mit der Hilfe von grafischen oder physischen Objekten können ebenfalls Programme erstellt werden [KP05]. Diese Objekte sollen Programmelemente oder einzelne Befehle darstellen, die miteinander kombiniert werden können [KP05]. Durch Formbeschränkungen der einzelnen Objekte wird verhindert, dass syntaktisch inkorrekte Befehle mit den Komponenten erzeugt werden [KP05]. Ein Beispiel hierfür ist Pict [KP05]. Mit Pict lassen sich einfache Programme durch das Verbinden von grafischen Bildern erstellen [KP05]. Die Bilder repräsentieren Befehle, die aus einer Palette in der Menüleiste ausgewählt werden können. Um dem Benutzer die Programmausführung visuell darzustellen, bewegt sich eine Box entlang der Befehlskette

und stoppt, sofern das Programmverhalten an dem bestimmten Punkt nicht spezifiziert ist [KP05]. Diese Visualisierung ist ein weiterer wichtiger Punkt, da viele Programmieranfänger noch wenig bis keine Vorstellung davon haben, wie ein Programm ausgeführt und durchlaufen wird [Sol86]. Ein Beispiel für eine datenflussbasierte visuelle Programmiersprache ist Show and Tell [KP05]. Diese Sprache wurde für Kinder entwickelt um sie näher an das Programmieren heranzuführen [KP05]. Ein Programm besteht hier aus Boxen die miteinander verbunden werden. Eine Box repräsentiert einen bestimmten Wert oder eine ausführbare Operation auf Werte [KP05]. Zusätzlich beinhaltet das Programm Boxen, die arithmetische Funktionen, Ein- und Ausgabemethoden und Spezialfunktionen wie das Abspielen von Musik erfüllen. Boxen können durch selbstgezeichnete Bilder markiert werden, um sie für spätere Verwendungen wieder zu erkennen [KP05]. Wie zu sehen ist, gibt es mehrere Wege, um Programme sinnvoll zu implementieren. So sind beispielsweise auch Systeme interessant, die bei der Strukturierung einer Programmiersprache mit einer visuellen Oberfläche unterstützen. Dennoch hängt es von der Wahrnehmung des Anwenders ab, welches das für ihn beste System ist.

PBD und visuelle Vorher-Nachher Regeln

Trotz der vielen Versuche Programmiersprachen einfacher zu gestalten oder eine komfortablere Befehlseingabe zu bieten, gelang es den Forschern nicht, eine wirklich breite Masse an Anfängern für das Programmieren zu begeistern [DCST00]. Laut einer Schätzung an der Universität von Michigan programmieren nur ca. 1 % der Teilnehmer eines Programmierkurs nach Ende der Vorlesung weiter. Um das Programmieren einfacher und interessanter zu gestalten, haben Wissenschaftler die Methoden PBD (Programing by demonstration = Programmieren durch Demonstrieren) und visuellen Vorher-Nachher Regeln kombiniert [DCST00]. Mit PBD werden Algorithmen von Benutzern, dem Computer vorgeführt. Das geschieht durch die Verwendung der Benutzeroberfläche mit einfachen Computer-Steuerungsgesten [DCST00]. Diese Steuerbefehle werden vom Computer aufgenommen und später bei der Programmausführung wiederholt [DCST00]. Durch diese Methode werden verwirrende sprachsyntaktische Elemente und komplexe Befehlseingabepraktiken umgangen. Zu beachten ist allerdings, dass nach dem Erstellen eines Programms, Veränderungen daran ebenfalls einfach gestaltet sein müssen. Diesen Punkt vergessen viele Systeme die PBD verwenden [DCST00]. Aus diesem Grund haben die Entwickler von Stagecast Creator [DCST00] die Beschränkung eingeführt, dass die Creator-Benutzer nur den Anfangs- und Endzustand, ohne die komplexen Zwischenschritte, eines Programms sehen. Möchte ein Benutzer also beispielsweise für einen Zugsimulator eine neue Regel definieren - der Zugmotor soll von links nach rechts bewegt werden - so definiert der Benutzer eine visuelle Vorher-Nachher Regel [DCST00]. D.h. der Vorher-Zustand wird definiert (Motor auf der linken Seite) und anschließend wird mit Drag-And-Drop der Motor auf die rechte Seite des Nachher-Zustand gezogen. Siehe Abbildung 2.1. Auf die gleiche Weise funktioniert auch das Programm KidSIM [1], mit welchem grafische Simulationen und Spiele erstellt werden können.

[1] http://www.sigchi.org/chi95/proceedings/papers/ac1bdy.htm

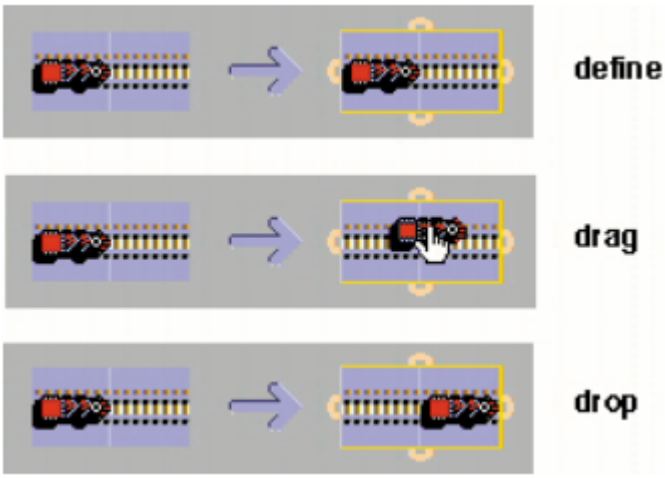

Abbildung 2.1.: PBD definieren einer Regel [DCST00]

2.6.2. Design einer Programmiersprache für Programmieranfänger

Basierend auf den Erkenntnissen der bereits entwickelten Lernsysteme für Programmieranfänger kann zusammengefasst werden, dass es viele Möglichkeiten gibt eine sinnvolle Entwicklungsumgebung für Programmieranfänger zu entwickeln. Die Hauptpunkte die dabei beachtet werden müssen, sind die Beseitigung der komplexen Sprachsyntax und die komfortable Eingabe der Befehle. Besonders die visuellen Lernsysteme überzeugen durch die Eliminierung von syntaktischen Beschränkungen, durch bereits bekannte physische Beschränkungen und lassen sich den Programmierer auf das Erstellen des Programms konzentrieren [KP05]. Zu beachten ist allerdings, dass mit textuellen Sprachen oft mächtigere Optionen zur Verfügung stehen, als durch rein visuelle Objekte (sofern sie sich nicht individuell gestalten lassen) [KP05]. Damit wäre eine Kombination aus visuellen Operationen und textuellen Eingabemethoden optimal. Ein gutes Beispiel sind die Programme StarLogo und NetLogo [2] [3], welche Multi-Agenten-basiert sind. Den Agenten werden durch programmieren von einzelnen Verhaltensfunktionen Regeln zugeteilt. Über die grafische Oberfläche lassen sich diese Agenten dann kombinieren, wodurch eine Simulation modelliert und ausgeführt werden kann.

Weitere Funktionen die in einer Entwicklungsumgebung für Anfänger gegeben sein sollten, sind zum Beispiel eine verständliche Programmausführung, evtl. visuell unterstützt und gute Rückmeldung bei auftretenden Fehlausführungen oder Konstruktionsfehlern. Ein gutes Beispiel und relativ erfolgreich an Probanden getestet, bietet das PBD Prinzip in Kombination mit den Vorher-Nachher Regeln [DCST00].

[2] http://education.mit.edu/starlogo/
[3] https://ccl.northwestern.edu/netlogo

3. Umfrage

Um einen Überblick über verwendete Simulationswerkzeuge im SimTech Cluster der Universität Stuttgart zu erlangen, haben wir eine Onlineumfrage erstellt. Diese Umfrage zielt außerdem darauf ab, zusätzliche Informationen zu der Verwendung dieser Werkzeuge zu erlangen und auch in welchem Fachgebiet diese hauptsächlich verwendet werden. Im folgenden Kapitel wird zuerst der Aufbau und die Entwicklung des Fragebogens präsentiert. Anschließend wird ein Blick auf die Durchführung der Umfrage geworfen. Am Ende des Kapitels werden die Ergebnisse des Fragebogens dargelegt.

3.1. Fragebogen

Dieser Abschnitt befasst sich speziell mit dem Fragebogen. Hier wird die Entwicklung und dessen Aufbau dargestellt. Wir starten mit den generellen Aufbau, erklären anschließend die Entwicklung des Fragebogens und gehen am Ende nochmal genauer auf die einzelnen Fragen und deren Hintergrund ein.

3.1.1. Genereller Aufbau

Dieser Abschnitt befasst sich mit dem groben Aufbau des Fragebogens (siehe Unterkapitel 3.1.3). Der Fragebogen ist aufgeteilt in vier Teile. Der erste Teil behandelt generelle Informationen über den Befragten. Dieser Teil ist optional da hier auch persönliche Daten wie Name und E-Mailadresse abgefragt werden. Der zweite Teil befasst sich mit der Art und Durchführung der Simulationen und Modellierungen, sowie die dafür verwendeten Programme. Der dritte Teil ist eine detaillierte Bewertung eines der verwendeten Programmen. Dieser Teil konnte für bis zu fünf Programme ausgefüllt werden. Der letzte Teil ist um das Einverständnis des Befragten zur Nutzung der Daten einzuholen und mögliche Vereinbarungen für abschließende Interviews zu finden.

3.1.2. Entwicklung

Die Entwicklung des Fragebogens lief über mehrere Schritte ab. Zuerst wurde eine Untersuchung durchgeführt um sich genauer mit Simulationen vertraut zu machen. Hierbei wurden gängige Werkzeuge genauer betrachtet, wobei das primäre Augenmerk auf deren Fachgebiet lag, um somit Programme zu finden, die möglicherweise auch von SimTech-Mitarbeitern in ihrem Arbeitsalltag verwendet werden. Dies erwies sich später als schwer, da es auch keinen wirklichen Überblick über alle Mitarbeiter und ihr Fachgebiet gab. Dadurch wurde auch nochmals deutlich wie vorteilhaft eine Liste ist, welche alle Mitarbeiter mit deren Tätigkeiten auflistet.

Aus den Ergebnissen der Untersuchung wurde ein erster Fragebogen erstellt. Dieser wurde zuerst intern genauer besprochen und danach einigen SimTech-Mitarbeitern aus unterschiedlichen Fachbereichen vorgelegt. Diese sollten den Fragebogen zur Probe ausfüllen und hatten die Möglichkeit, eine Bewertung abzugeben bzw. Änderungsvorschläge zu machen. Aus den Interviews wurde bekannt, wie unterschiedlich die Kenntnisse der Mitarbeiter im Bereich der Begrifflichkeiten und auch die Art und Weise, wie diese Simulieren bzw. Modellieren, sind.

Anhand der Resultate aus den Interviews wurde ein weiterer Fragebogen erstellt. Dieser wurde dann endgültig mithilfe der Umfrageplattform LimeSurvey [1] digitalisiert. Hierbei wurden noch kleinere Änderungen durchgeführt.

3.1.3. Fragen

Dieser Teil behandelt die Fragen im Detail. Dabei werden vor allem die Hintergründe und der erwartete Nutzen der einzelnen Fragen genannt und erklärt. Für eine übersichtlichere Darstellung werden die einzelnen Teile des Fragebogens getrennt betrachtet. Wir beginnen mit den generellen Fragen um anschließend die Fragen zu Modellierungs- und Simulationsprozessen zu behandeln. Danach wird die Fragegruppe zur Werkzeugbewertung erläutert. Am Ende wird ein kurzer Blick auf die Fragen zum Einverständnis geworfen.

Generelle Fragen

Dieser Teil behandelt generelle Fragen, die zur Erfassung von Beschäftigung und Zugehörigkeiten zu Abteilungen von den Teilnehmer gedacht ist. Alle Fragen in dieser Gruppe waren optional.

Zuerst wird nach Name und E-Mailadresse gefragt, da es sich hier um simple Datenerfassung handelt. Dies dient zur Erfassung der Teilnehmer, um somit die Bandbreite der Teilnehmer zu ermitteln. Die Frage nach dem Projektnetzwerk und dem Fachbereich dient dazu, die Mitarbeiter zu gruppieren. Dadurch können statistische Aussagen, über die Verwendung von bestimmten Programmen, getroffen werden. Weiterhin haben diese Fragen auch den Zweck genauere Daten über den Mitarbeiter zu erfassen.

Sollte ein Teilnehmer bei der Frage nach dem Projektnetzwerk angeben, dass er kein SimTech-Mitarbeiter ist, so wird ihm die Möglichkeit geboten seine Universität anzugeben. Der Fragebogen wurde um diese Möglichkeit erweitert, um auch Simulationsexperten, die nicht bei SimTech arbeiten, die Chance zu geben an der Umfrage teilzunehmen. Damit sollte auch der Befragtenkreis erhöht werden, um eine größere statistische Aussagekraft der Ergebnisse zu erhalten.

[1] http://www.limesurvey.org

Modellierung- und Simulationsprozess

Dieser Teil zielt darauf ab, die Vorgehensweisen und verwendeten Methoden der Teilnehmer herauszufinden. Dabei sind außerdem verwendete Programme und Ansätze zur Simulation anzugeben.

Um Aufgabenbereiche und Tätigkeiten zu ermitteln, fragen wir nach Simulationsschritten und einer groben Beschreibung der Vorgehensweise. Es ist in soweit relevant, da in unterschiedlichen Prozessschritten unterschiedliche Werkzeuge verwendet werden bzw. auch die Werkzeuge in diversen Schritten zu unterschiedlichen Zwecken verwendet werden. Diese Frage wurde durch die vorangegangenen Interviews geprägt, da uns hierbei erklärt wurde, dass beim Modellieren/Simulieren auch noch das Sammeln von Daten, die Visualisierung und das Interpretieren der Daten relevant ist.

Bei der Frage nach den Prozessschritten galt es herauszufinden, ob Personen, die in den selben Prozessschritten arbeiten, unterschiedliche Werkzeuge verwenden. Dies kann auf Grund der Zugehörigkeit zu unterschiedlichen Bereichen der Fall sein. Außerdem galt es auch zu ermitteln ob Werkzeuge in mehreren Prozessschritten verwendet werden.

Des weiteren werden die Teilnehmer nach einer Klassifikation bzw. Einschätzung der verwendeten Simulationsansätze gefragt. Dabei soll ermittelt werden, ob immer die selben Werkzeuge bei z.B. einer statischen Simulation verwendet werden oder ob diese auch für andere Simulationen Anwendung finden.

Außerdem werden bekannte Programme abgefragt, um zu ermitteln, ob in den selben Gebieten (diese können der Prozessschritt, der Ansatz oder auch das Fachgebiet sein) die gleichen Werkzeuge bekannt sind, oder ob es große Variationen gibt. Durch die vorangegangenen Interviews wurde ersichtlich, dass es viele Mitarbeiter ihre Werkzeuge selbst schrieben. Aus diesem Grund wurde die Frage nach diesen Werkzeugen hinzugefügt.

Zuletzt werden die Teilnehmer gebeten, ihre eigene Programmiererfahrung einzuschätzen. Mit Hilfe dieser Angabe sollten genauere Aussagen über die später genannte Programmiererfahrung für die einzelnen Programme getroffen werden.

Werkzeugbewertung

Die ersten Fragen in diesem Block hatten den Zweck herauszufinden, welches Werkzeug zu welchem Zweck in welchem Kontext verwendet wird.

Bei den nächsten Fragen war das Ziel herauszufinden wie das Programm genutzt wird. Dabei waren die primäre Eingabemethode anzugeben, um später Aussagen darüber zu treffen ob z.B. eine grafische Benutzeroberfläche der Allgemeinheit besser gefällt als der Nutzen von Programmcode. Außerdem soll die ermittelt werden, welche Eingabemethoden das Programm unterstützt und wie beliebt diese sind.

Darauf folgte ein Einschätzung bezüglich der Nutzerfreundlichkeit, welche aus einigen Fragen aus dem Computer System Usability Questionnaire (CSUQ) (siehe Anhang A.2.4) besteht. Ziel dabei ist etwas über die Bedienbarkeit des Werkzeugs herauszufinden, um abschätzen zu können, ob ein

Werkzeug besser und Nutzer freundlicher ist. Einige Fragen aus dem vollständigen CSUQ wurden hier verworfen, da sonst der Fragebogen zu lang geworden wäre.

Die Fragen nach fehlenden Funktionen hatten das Ziel herauszufinden, ob es bei dem Werkzeug einen großen Malus gibt. Dadurch soll fehlende Funktionen ermittelt werden, die für zukünftige Programme wichtig sind.

Mit den folgenden Fragen galt es herauszufinden, wie lange der Teilnehmer schon mit dem Werkzeug arbeitet und wie intensiv die Nutzung ist. Dabei soll eine Einschätzung der Expertise für dieses Programm ermittelt werden.

Zusätzlich wird nach Programmiersprachen gefragt, die zur erfolgreichen Nutzung des Werkzeugs benötigt werden. Das soll bei unserer Einschätzung der nötigen Programmiererfahrung helfen.

Am Ende soll der Teilnehmer selbst noch eine Einschätzung zu mehreren Dingen abgeben. Dabei ist die Bedienung des Programms, der manuelle Programmieraufwand und die Beherrschung der relevanten Sprachen von Bedeutung. Damit sollen die vorangegangenen Fragen hervorgehoben und eine Einschätzung unsererseits ermöglicht werden.
Zuletzt fragen wir nach einem integrierten Editor. Dabei soll die Unterstützung in programmieraufwendigen Werkzeugen ermittelt werden.

Einverständnisserklärung

Die Letzte Fragegruppe des Fragebogens behandelt das Einverständnis der Teilnehmer. Mit der Frage nach dem Einverständnis zur Veröffentlichung wollten wir sicher gehen, dass auch ein Mehrwert für die Mitarbeiter von SimTech erreichbar ist. Zusätzlich wurden die Teilnehmer gefragt, ob sie über die Ergebnisse der Umfrage informiert werden wollen. Die Teilnehmer konnten außerdem angeben, ob sie uns einen Einblick in ihre Arbeit gewähren wollen. Dabei wollten wir Eindrücke von den verwendeten Programmen und den nötigen Schritten zur erfolgreichen Durchführung einer Simulation erlangen. Zuletzt hatten die Teilnehmer noch die Chance, Kommentare jeglicher Art in einem Freitextfeld anzugeben.

3.2. Durchführung

Wir haben eine Onlineumfrage auf der Plattform LimeSurvey erstellt. In dieses wurden die Fragen eingegeben und durch Beschreibungstexte erweitert. Die Teilnehmer wurden per E-Mail aufgefordert an der Umfrage teilzunehmen.

Im Einleitungstext wurde den Teilnehmern das Thema vorgestellt. Anschließend sollte der Teilnehmer die Fragen in der oben angegebenen Reihenfolge ausfüllen. Dabei hatte jeder Teilnehmer die Möglichkeit, eine Bewertung für bis zu fünf Programme abzugeben.

3.3. Teilnehmer

Insgesamt hatte die Umfrage 46 Teilnehmer. Davon haben fünf angegeben, nicht zum SimTech Cluster der Universität Stuttgart zu gehören. Mit einer Anzahl von 28 Teilnehmern war die Gruppe der Doktoranden am stärksten unter den Teilnehmern vertreten.

3.4. Ergebnisse

Die Fragen Name und E-Mailadresse werden hier nicht gezeigt, da dies eine liste von Namen und E-Mailadressen ist, diese können in der Liste im Anhang gefunden werden.

3.4.1. Generelle Fragen

Von den Befragten waren 28 Teilnehmer Doktoranden, vier Doktoren, acht Professoren und vier gaben an, einer anderen Beschäftigung zu folgen. Die Angaben bei anderen Beschäftigungen setzen sich aus drei Juniorprofessoren und einem Student zusammen. Zwei Teilnehmer gaben in diesem Feld nichts an (siehe Abbildung 3.1).

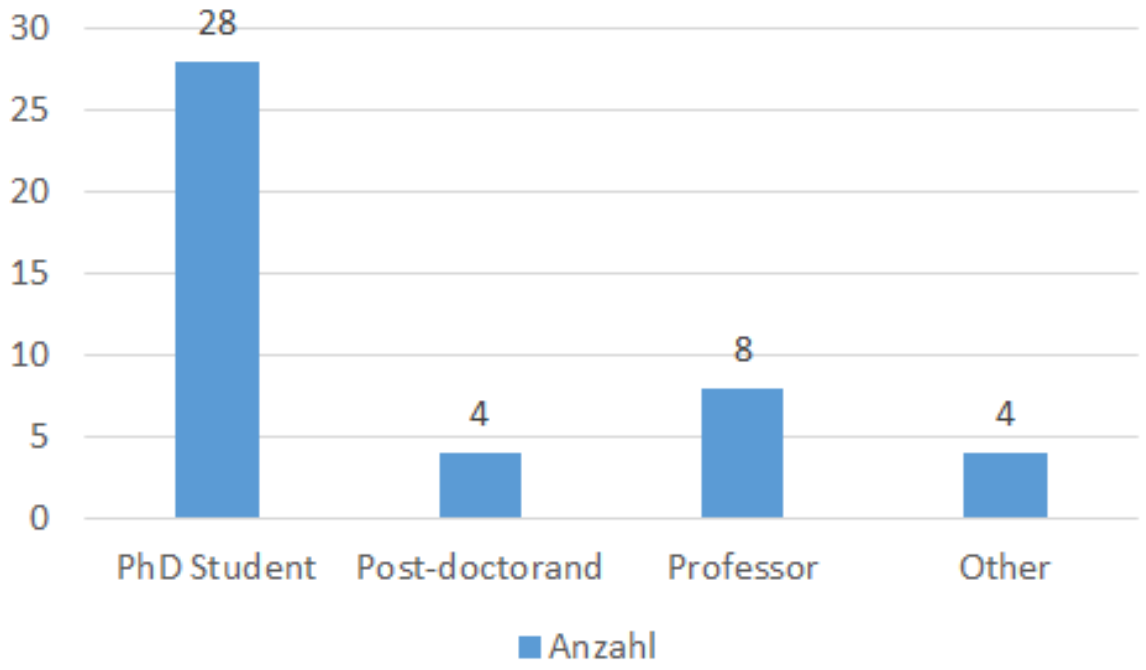

Abbildung 3.1.: Die Verteilung der Teilnehmer in Abhängigkeit der Beschäftigung

Von den Befragten gaben insgesamt 15 Teilnehmer ein Diplom, zwölf einen Master, neun einen Doktortitel, sechs eine Habilitation, einen Bachelortitel und ein Staatsexamen als höchsten Abschluss an. Die verbleibenden zwei Teilnehmer entschieden sich, diese Frage nicht zu beantworten.

Von den Teilnehmern kamen aus jedem Projektnetzwerk durchschnittlich fünf Befragte, wobei neun aus PN4: Gekoppelte Probleme in Biomechanik und Systembiologie und nur einer aus PN6: Wege zu intelligenten Simulationsinfrastrukturen kamen. Außerdem gaben zwei Teilnehmer an, nicht bei SimTech angestellt zu sein. Von diesen gab jeweils ein Teilnehmer an dem Imperial College London oder der School of Mathematics University of Edinburgh zugehörig zu sein. Außerdem gaben fünf Teilnehmer keine Antwort auf diese Frage an (siehe Abbildung 3.2).

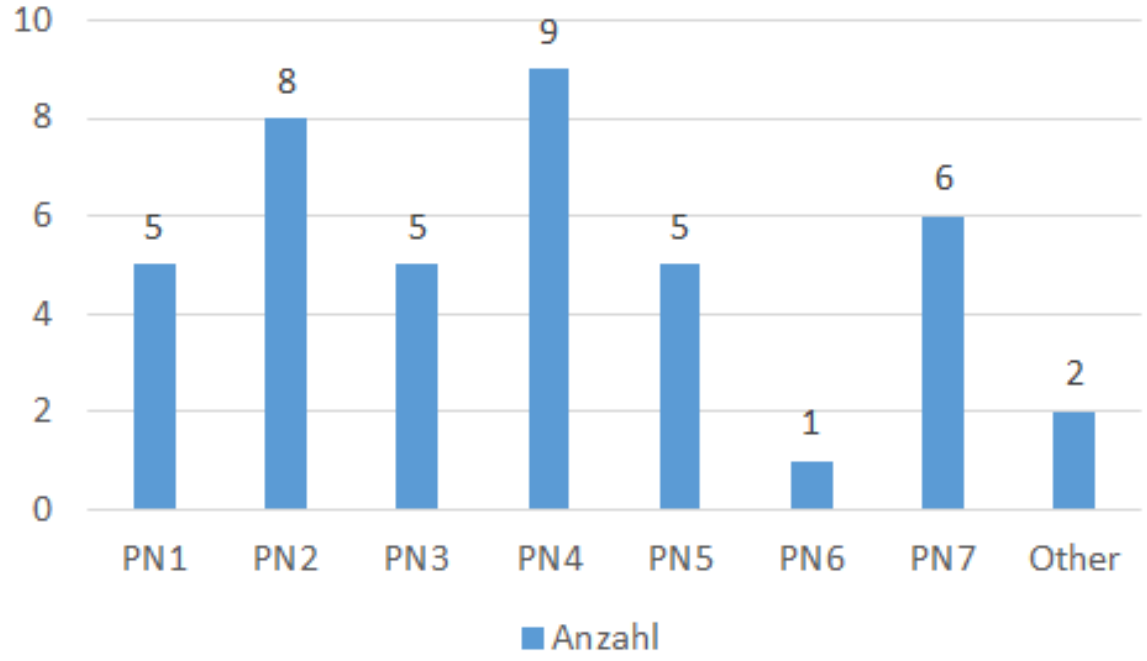

Abbildung 3.2.: Die Zugehörigkeit der Teilnehmer zu den Projektnetzwerken des SimTech Clusters.

Die Fachrichtung setzte sich zum größten Teil aus Ingenieuren (41.3%) und Computerwissenschaftlern (17.4%) zusammen (siehe Abbildung 3.3). Weniger stark waren Mathematik (10.9%) und andere Naturwissenschaften (13.0%) vertreten. Als andere Fachrichtung gaben zwei Teilnehmer Physik und jeweils ein Teilnehmer Biomechanik an.

3.4.2. Modellierung- und Simulationsprozess

Die meist genannten Schritte, an denen gearbeitet wird, sind Simulation mit 27 und Modellierung mit 27 Angaben (siehe Abbildung 3.4). Als weiterer großer Teil wurde die Interpretation mit 17 Angaben ausgewählt. Sammeln (8 Angaben) und Visualisieren (10 Angaben) machen nur ein kleiner Teil der Angaben aus. Als zusätzlichen Schritt wurde “Optimization-Inverse Modelling” angegeben.

Es wurden insgesamt 44 unterschiedliche Angaben gemacht, die die Arbeit in den einzelnen Schritten genauer spezifiziert. Da diese Angaben sehr speziell sind und nur schwer zu gruppieren, verzichten wir an diesem Punkt auf eine geeignete Zusammenfassung und verweisen auf die komplette Liste der Antworten im Anhang (siehe Anhang A.2.1).

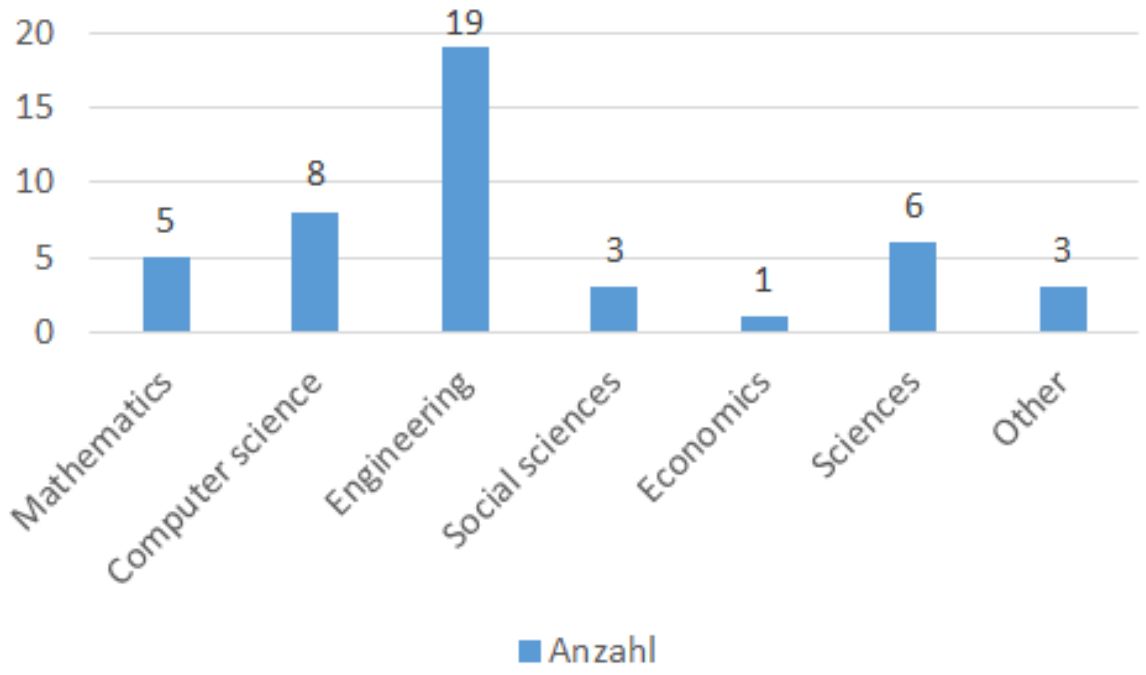

Abbildung 3.3.: Die Fachrichtungen der Teilnehmer

Der meist genannte Ansatz war unterschiedliche Arten Differenzialgleichungen zu simulieren, wobei am häufigsten zwischen PDE (partial differential equations) und ODE (ordenary differential equations) unterschieden wurde. Von 28 Simulationen zu Differenzialgleichungen wurden 20-mal PDE, sechsmal ODE und zweimal andere genannt. Dabei wurde PDE zum Teil noch genauer als FEM (finite element method) beschrieben. Ein weiterer Simulationsansatz der häufiger genannt wurde war Molecular Dynamics (sechsmal). Außerdem wurde auch Monte Carlo (dreimal) häufiger als Simulationsansatz genannt. Es wurden noch neun weitere Ansätze genannt, wobei hier nur fünf Modellierungsansätze genannt wurden, wie z.B. Agenten basierte Modellierung. Des weiteren konnten vier Ansätze weder Modellierung, noch Simulation zugeordnet werden.

Bei der Klassifizierung (siehe Abbildung 3.5) ist der Vergleich der Gegensatzpaare Relevant (siehe Kapitel 2.3). Das erste Paar ist statisch und dynamisch. Die Teilnehmer gaben mehr als doppelt so oft dynamisch (32-mal) als statisch (14-mal) an. Das zweite Gegensatzpaar ist kontinuierlich und diskret. Die Teilnehmer haben in diesem Fall eine ähnliche Verteilung, mit kontinuierlich (22-mal) und diskret (17-mal), angegeben. Das letzte Gegensatzpaar, deterministisch und stochastisch, variiert nur um zwei Werte. Deterministisch wurde 17-mal angegeben und Stochastisch 15-mal. Als Anderes wurde “Qualiataive impact network analysis”, einmal “none” und einmal “n/a” angegeben.

Bei der Frage nach den meist bekannten Werkzeugen wurden insgesamt 76 unterschiedliche Programme genannt (siehe Anhang A.2.2). 61 Programme wurden nur von einem Teilnehmer erwähnt. Matlab war das am häufigsten genannte Programm, mit insgesamt 15 Nennungen. Das zweit meist genannte Programm war OpenCMISS mit insgesamt fünf Erwähnungen. Die nächst häufig genannten Programme sind DUNE und ESPResSo mit vier Angaben. Weitere neun Programme wurden jeweils zweimal

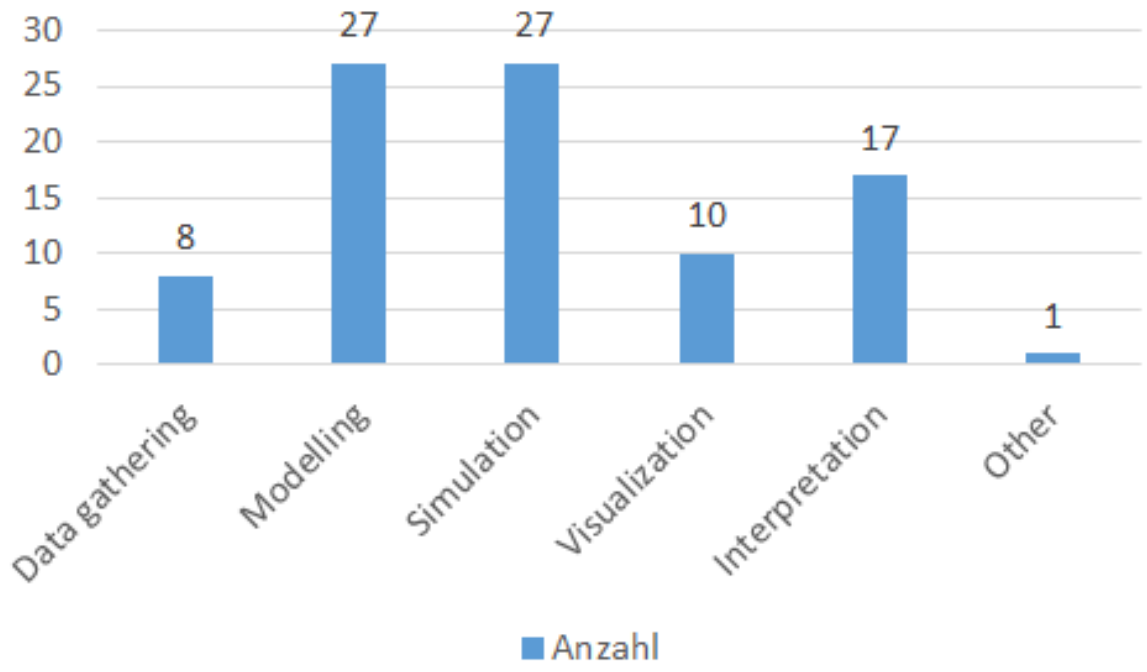

Abbildung 3.4.: Die Schritte an denen von die Teilnehmer arbeiten

genannt. Diese Programme sind DUMUX, ParaView, GROMACS, Maple, preCICE, OpenFOAM, COMSOL, LAMMPS und Eclipse, wobei Eclipse eine Entwicklungsumgebung ist. Zusätzlich wurden mit C++ (viermal), C (einmal) und Fortan (einmal) auch einige Programmiersprachen genannt.

Die gesamte Anzahl der für die Arbeit genannten Programme (siehe Anhang A.2.3) variiert nicht groß mit den im Feld bekannten Programmen. Hier wurde eine Anzahl von 73 unterschiedlichen Programmen genannt, wobei 53 nur einmal genannt wurden. Abermals ist hier das meist genannte Programm Matlab mit 13, gefolgt von DUMUX und OpenCMiss mit jeweils vier Auflistungen. Programme die dreimal genannt wurden sind DUNE, SG++ und ESPResSo. Dreizehn weitere Programme wurden jeweils zwei mal genannt. Eine der Ausnahmen die nicht unter die Kategorie Simulation- und Modellierungsprogramm fallen, ist Linux als Betriebssystem mit insgesamt zwei Nennungen. Weiterhin wurden vier Textverarbeitungsprogramme und Präsentationsprogramme genannt. Diese sind Latex, Powerpoint, Excel und Word. Außerdem wurden C++ (viermal), Python (zweimal) und Fortran als Programmiersprachen genannt. Des weiteren wurden auch die Entwicklungsumgebungen bzw. Texteditoren Sublime und Eclipse (zweimal) genannt. Als letzte Kategorie wurden Programme zur Versionsverwaltung genannt. Diese sind git mit zwei und SVN mit jeweils einer Nennung. Insgesamt 33 von 47 Teilnehmern, die diese Frage beantwortet haben, gaben an, ihre Programme selbst zu programmieren, bzw. verwenden ein vom Institut entwickeltes Programm.

Die Programmierfähigkeiten der Teilnehmer wird im Schnitt mittelmäßig bis gut bewertet. Der meist angegebene Wert ist gute Programmierfähigkeiten mit 16 Angaben. Zwei Teilnehmer gaben sehr schlechte Programmierfähigkeiten an (siehe Abbildung 3.6).

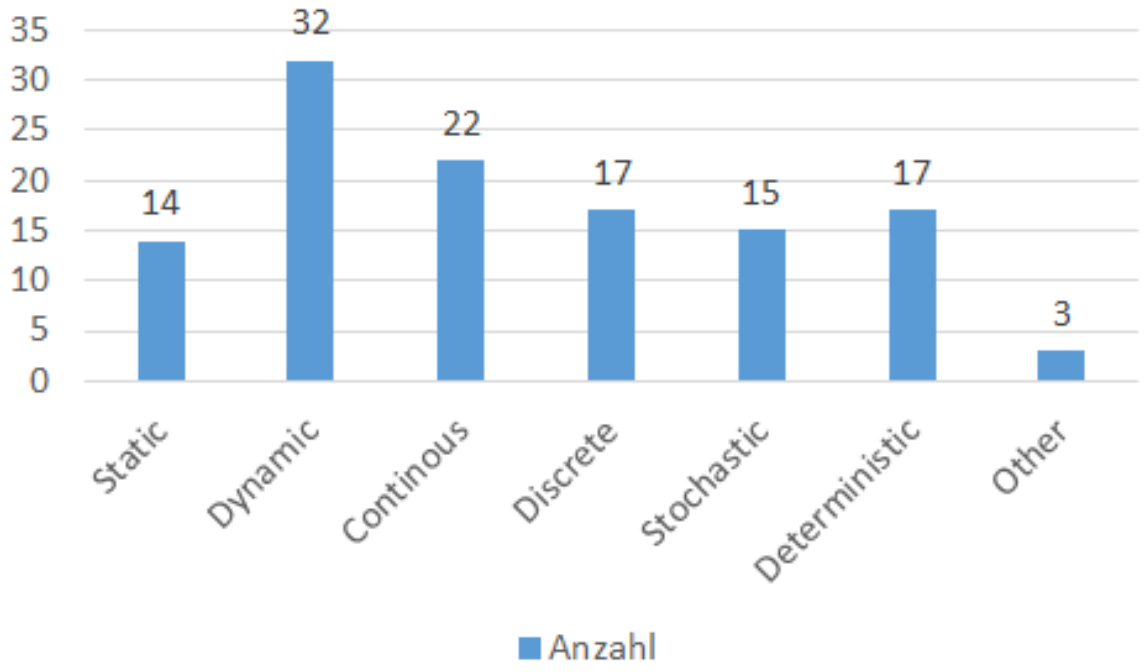

Abbildung 3.5.: Klassifizierung der Arbeit der Teilnehmer

3.4.3. Programm Bewertung

Es wurden insgesamt 27 Programme bewertet. Davon wurden insgesamt 23 Programme nur einmal bewertet und werden in diesem Kapitel nicht näher erläutert (siehe Anhang A.2.4 für die komplette Liste). Im folgenden werden also die Bewertungen von Programmen genauer betrachtet, welche mindestens zweimal bewertet wurden (für CSUQ-Details siehe Anhang A.2.4).

Das erste Programm mit drei Bewertungen ist ESPResSo. Dieses Programm wird von allen Teilnehmern zum Simulieren und von einem Teilnehmer zusätzlich zum Modellieren verwendet. Zur Bedienung ist keine grafische Oberfläche vorhanden, sondern wird mit Hilfe von Programm-spezifischem bzw. Externen Code bedient. Die Bedienungsfreundlichkeit aus dem CSUQ ergab sich ein gesamt Durchschnittswert von 3,54, wobei hier kein Aspekt sowohl negativ als auch positiv herausragt. Als gewünschte Verbesserungen wurden Algorithmen für moderne Physik, Python Interface, Paralleles Input/Output und eine generelle Fehlerbehebung genannt. Dabei wünscht sich mehr als die Hälfte der Teilnehmer, die das Programm bewertet haben, ein Python Interface und parallelen Input/Output. Zwei der Teilnehmer verwenden das Programm zwischen zwei und fünf Jahren, während ein Teilnehmer seit über 5 Jahren mit ESPResSo arbeitet. Das Programm wird durchschnittlich 57% der Arbeitszeit verwendet, wobei das Minimum bei 10% liegt. Zur Verwendung wird primär eine Programm Kommando Sprache (TCL) verwendet, aber auch C, C++, Message Passing Interface (MPI) und Python sind möglich. Zwei Teilnehmer können das Programm sehr gut bedienen und einer durchschnittlich gut. Zwei Teilnehmern gaben an, dass gute Programmierkenntnisse benötigt werden und laut einem

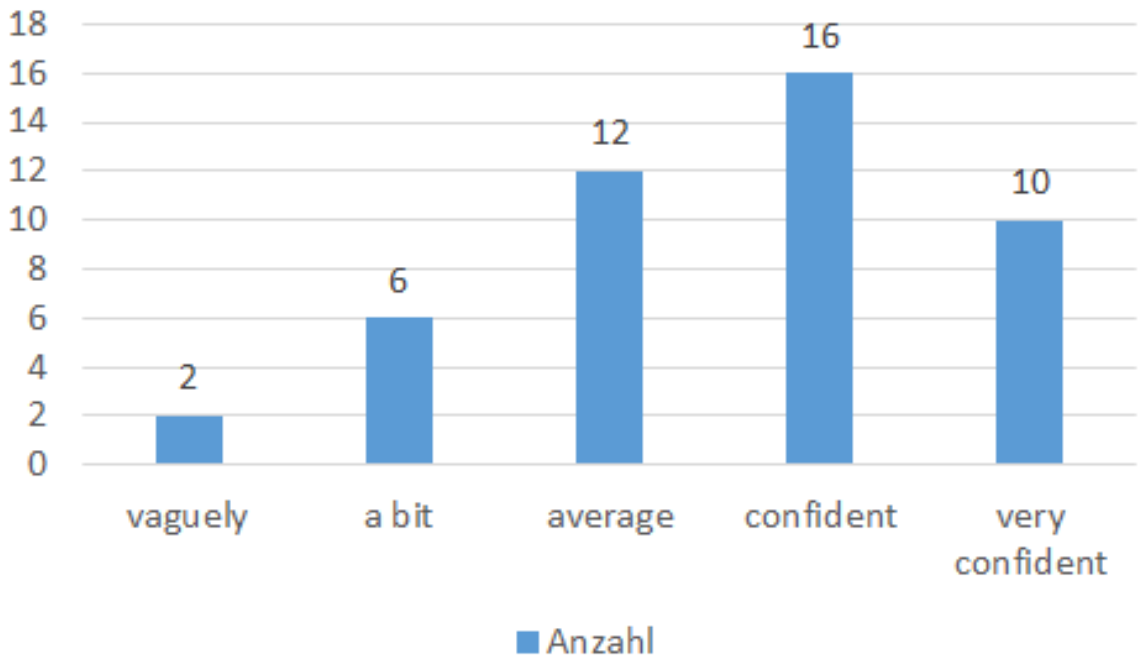

Abbildung 3.6.: Die Programmierfähigkeiten der Teilnehmer

sind weniger gute Programmierkenntnisse vorausgesetzt. Die Teilnehmer beherrschen die benötigten Programmiersprachen im Schnitt gut. Das Programm liefert keinen integrierten Editor mit.

Das nächste Programm ist (Open)CMISS das von insgesamt vier Teilnehmern bewertet wurde. Dieses Programm wird von allen Teilnehmern zum sowohl Simulieren als auch Visualisieren verwendet. Zusätzlich verwenden es drei Teilnehmer zur Modellierung, zwei zur Daten Sammlung und ein Teilnehmer zum Interpretieren. Genauso wie ESPResSo wird (Open)CMISS über Programm-spezifischem bzw. Externen Code bedient. Aus dem CSUQ ergab sich eine durchschnittliche Bedienungsfreundlichkeit von 2,8, wobei hier die Aspekte der Erlernbarkeit mit 1,5 negativ und die allgemeine Zufriedenheit mit 3,75 positiv hervorragen. Als Mängel von (Open)CMISS werden fehlende Beispiele für den Anfang, ein klarer Quellcode und neue Features wie z.B. Verlinkung zu einem GUI genannt. Ein Teilnehmer verwendet das Programm seit ein bis zwei Jahren, zwei verwenden es bisher zwischen zwei und fünf Jahren, während ein weiterer (Open)CMISS seit über fünf Jahren verwendet. Der Beanspruchung der Arbeitszeit variiert von zwei bis 50% der Arbeitszeit, wobei der Schnitt bei 25,5% der Arbeitszeit liegt. Als benötigte Programmiersprache wurde Fortran angegeben und das Programm setzt gute Fortran Kenntnisse voraussetzt. Die Fortran Kenntnisse der Teilnehmer werden als durchschnittlich angegeben. Jeweils ein Teilnehmer kann das Programm gut bzw. weniger gut und zwei können es durchschnittlich bedienen. (Open)CMISS liefert keinen integrierten Editor mit.

Ebenso wie (Open)CMISS wurde auch das Programm DUNE von vier Teilnehmern bewertet. Das Programm wurde von allen Teilnehmern zum Simulieren verwendet. Außerdem verwenden es zwei der Teilnehmer zur Modellierung und ein Teilnehmer zum Visualisieren. Auch DUNE wird mit Hilfe von Programm-spezifischem bzw. externen Code bedient. Die durchschnittlich angegebene

Bedienungsfreundlichkeit beträgt 3,08, wobei die Aspekte der Simplizität des Programms mit einem Wert von 1,25 negativ und die Effizenz mit 4,5 positiv hervorragen. Gewünschte Verbesserungen sind eine breitere Auswahl an Lösungsalgorithmen für Lineare Gleichungssysteme, eine verbesserte Spezifikation der parallelen Schnittstelle, zusätzliche Module und bessere Einführung für Personen mit geringeren C++ Fähigkeiten. Die Verwendungsdauer des Programms variiert stark. Es gibt einen Teilnehmer der das Programm seit über fünf Jahren verwendet und einen der erst seit maximal einem Jahr damit arbeitet. Es wird durchschnittlich 53% der Arbeitszeit mit DUNE verbracht, wobei das angegebene Minimum bei 30% liegt. Zur Verwendung werden gute bis sehr gute C++ Kenntnisse als Voraussetzung angegeben. Drei der Teilnehmer haben gute Kenntnisse in C++ und einer weniger gute. Die Teilnehmer gaben eine gute Expertise für das Programm an, wobei ein Teilnehmer das seine Expertise eher gering einschätzt. Das Werkzeug liefert keinen integrierten Editor mit.

Das letzte und meist bewertete Programm ist Matlab mit insgesamt neun Bewertungen. Matlab wird von allen zum Simulieren verwendet. Acht der Teilnehmer verwenden das Programm auch zum Visualisieren und sechs zum Modellieren. Außerdem wird Matlab von fünf Teilnehmern zum Interpretieren und von zwei zum Daten Sammeln verwendet. Zur Bedienung von Matlab wurde sowohl die Kommandozeile als auch Programm-spezifischer Code angegeben. Die angegebene allgemeine Bedienbarkeit von Matlab beträgt 4,08. Keine der einzelnen Attribute des CSUQ weicht seht deutlich von dem allgemeinen Durchschnittswert ab. Es wurden zwei Verbesserungsvorschläge gemacht, das ein besserer Editor und Dokumentation von einfachen Funktionen hinzugefügt werden. Aber sieben der Teilnehmer haben keine Verbesserungsvorschläge für Matlab gemacht. Zwei der neun Teilnehmer benutzen Matlab zwischen ein und zwei Jahren, vier benutzen es bereits zwischen zwei und fünf Jahren und drei länger als fünf Jahre. Matlab wird von den Teilnehmern durchschnittlich 33% der Arbeitszeit verwendet. Zur Bedienung von Matlab wird laut 4 Teilnehmern die Programmiersprache Matlab benötigt und 5 Teilnehmer gaben keine benötigte Programmiersprache an. Die durchschnittlich benötigte Programmiererfahrung wird als mittelmäßig eingestuft, wobei zwei Teilnehmer angegeben haben, dass sehr gute Programmiererfahrung benötigt wird. Sieben der Teilnehmer beherrschen die für Matlab benötigten Kenntnisse gut bis sehr gut, zwei mittelmäßig. Vier der Teilnehmer schätzen ihre Expertise für Matlab als gut ein, zwei als sehr gut, zwei weitere als weniger gut und ein Teilnehmer als durchschnittlich.

3.5. Diskussion

Aus den Ergebnissen der Umfrage haben sich einige Dinge klar heraus gestellt. Zum einen ist der SimTech Exzellenzcluster ein Zusammenschluss aus Experten der unterschiedlichsten Fachrichtung. Dabei unterscheidet sich nicht nur die Fachrichtung, sondern auch die genauen Anwendungsfälle und Vorgehensweise jedes Mitglieds. Das macht es zu einer großen Herausforderung, ein einheitliches Programm für den komplette SimTech Cluster zu entwickeln. Außerdem heben die Angaben der verwendeten Modellierungs- und Simulationsansätze deutlich hervor, dass für die selben Ansätze oft völlig unterschiedliche Methoden, Programme und Vorgehensweisen von Nöten sind. Da einige Teilnehmer Programmiersprachen, Entwicklungsumgebungen und andere, nicht speziell für die Simulation oder Modellierung entwickelte, Programme angegeben haben, ist der Anteil der selbst entwickelten Simulationen recht groß. Dabei ist davon auszugehen, dass diese Teilnehmer die Modelle

oder Simulationen entweder direkt mit Programmcode verwirklichen, oder selbständig Programme zur Simulation entwickeln.

Die 73 genutzten Programme zeigen die extreme Bandbreite an verfügbaren Lösungen für Simulationen. Dabei wird aus den vielen speziell entwickelten Programmen ersichtlich, dass auch nur wenig kommerzielle Lösung den nötigen Funktionsumfang aufweisen. Die Ergebnisse verdeutlichen außerdem, dass teilweise eine Ausführung von mehreren Programmen hintereinander zur Lösung der Aufgabe verwendet werden muss. Im Bezug auf nötige Programmiererfahrung und Komfort der bewerteten Programme, sind sich die Teilnehmer der Studie uneinig. Während manche Teilnehmer angaben, dass keine Programmiererfahrung zur Verwendung eines Programms nötig ist, gaben andere beim selben Programme nötige Programmiererfahrung an. Dies verdeutlicht, dass es viele unterschiedliche Anwendungsszenarien für die einzelnen Programme gibt. Außerdem lässt sich daraus schließen, dass einige Teilnehmer nur die bereits implementierten Funktionen der Programme benötigen, während andere Teilnehmer speziellere Funktionen benötigen und diese eventuell selbst schreiben müssen.

Außerdem sind uns Zusammenhänge zwischen nötiger Programmiererfahrung und Erlernbarkeit eines Programmes aufgefallen. Je höher die nötige Programmiererfahrung für ein Programm, umso schlechter wir die Erlernbarkeit des Programms bewertet. Zusätzlich ist unter den mehrfach bewerteten Programmen keines dabei, das eine grafische Benutzeroberfläche anbietet, darum wird auch die Programmiererfahrung für diese Programme durchschnittlich als hoch empfunden. Bei diesen Programmen gibt es außerdem keine einheitlich verwendete Programmiersprache. Es werden sehr unterschiedliche Sprachen wie Matlab, C++ und Fortran genannt.

4. Interviews

In der Umfrage gibt es die Möglichkeit anzukreuzen, ob Teilnehmer damit einverstanden wären uns einen Einblick in deren Arbeit mit den angegeben Simulationswerkzeugen zu gewähren. Die daraus resultierenden Ergebnisse sollten uns weitere Informationen, über die Anwendung der angegebenen Programme geben. Wir haben Fragen vorbereitet, mit denen wir diese Informationen herausfinden können. Neben der Frage nach einer kurzen Demonstration der Prozessausführung, fragten wir auch warum das Programm auf diese Weise und nicht anders genutzt wird, bzw. ob es auch anders genutzt werden kann. Des Weiteren haben wir nach dem persönlichen Eindruck über die verwendeten Werkzeuge gefragt. D.h. was gefällt dem Benutzer besonders an dem Programm und ob es kurz demonstriert werden kann. Außerdem wollten wir erfahren ob es Möglichkeiten der Erweiterung des Programms gibt, z.B. durch das Integrieren von Addons oder Plugins. Während der Demonstration haben wir uns Notizen zu der Bedienung des Programms gemacht, um selbst beurteilen zu können ob das Programm komfortabel zu benutzen ist.

Interviews haben wir mit Experten aus PN 5: Multiphasen- und Multiphysikmodellierungen geführt, welche in den Bereichen Ingenieurswesen und Mathematik tätig waren. Dabei wurden Partikelsimulationen und Durchfluss von Flüssigkeiten durch poröse Medien als Aufgabenbereiche angegeben.

In den Interviews haben wir herausgefunden, dass hauptsächlich Programme verwendet werden die am Institut entwickelt wurden. Diese Programme werden dann teilweise von den anwendenden Personen um Funktionen erweitert. Da diese zusätzlichen Funktionen meist nur von Einzelpersonen verwendet werden, kommen die Funktionen nicht in Umlauf und sind somit anderen Personen des Instituts nicht bekannt. In einem Simulationsprozess werden laut den befragten Personen mehrere verschiedene Programme benutzt. Einer der Teilnehmer nannte im Interview einen “riesen Klotz” an Daten, den er in die “Tool-Chain” eingeben muss. Das bedeutet ihnen steht kein allgemeines Werkzeug zur Verfügung, um Simulationen zentralisiert zu erstellen und durchzuführen. Oftmals werden Dateien mit einem Programm erzeugt, die dann anschließend von weiteren Programmen interpretiert und verarbeitet werden. Auf Nachfrage ob den Personen ein generisches Programm helfen würde, bekamen wir eine positive Antwort. Ein Interviewteilnehmer gab an, dass ein Bedarf für ein allgemeines Simulationswerkzeug im SimTech Cluster besteht.

Als komfortable Funktionen der verwendeten Programme, nannten befragte Personen, eine “visuelle Ausgabe von Simulationsergebnissen”. So seien Ergebnisse einer Simulation besser nachvollziehbar. Ein weiterer wichtiger Punkt ist laut den befragten Personen, eine informative Fehlerausgabe (“[...] kann direkt auf den Fehler klicken und hingehen [...]”). Dadurch ließen sich Fehlkonstruktionen schnell beheben. Um die Simulationswerkzeuge schnell bedienen zu können wurde genannt, dass Tastenkürzel gerne verwendet werden, da sich damit die Arbeitsprozesse beschleunigen lassen. Es wurde außerdem erwähnt, dass Programme ab einer bestimmten Anzahl oder Größe von Eingabedateien langsamer werden, was den Arbeitsfluss störe (“Bei großen Daten funktioniert Paraview nichtmehr so gut [...]”).

Teilweise müssen bei Programmen pro Berechnung alle Ausführungsparameter neu definiert werden. Das wurde bei einem Programm, dessen Benutzerfreundlichkeit nicht so gut sei, als negativer Punkt bemängelt. Der Mangel an Benutzerfreundlichkeit sei allerdings der Komplexität des Programms zu verschulden. Eine Person gab an, dass es allgemein schwer sei, eine Balance zwischen Komplexität und Benutzerfreundlichkeit zu finden ("[...] sehr komplex, deshalb ist die Balance zwischen Usability und Komplexität schwer."). In diesem Fall sei eine gute Dokumentation bei Simulationswerkzeugen erwünscht, so die befragte Person.

5. Empfehlung

Das SimTech Cluster besteht aus unterschiedlichen Gebieten, die jeweils unterschiedliche Modellierungsansätze und Simulationsprogramme benötigen. Deshalb sind wir der Meinung, dass ein geeignetes Simulationswerkzeug einen generischen Ansatz verfolgen muss. Programme die diagrammbasierte Modellierungsansätze verwenden, konnten sich für diese Herausforderung als besonders geeignet beweisen. Das Prinzip der Blockdiagrammnotation als Modellierungsansatz macht einen sinnvollen Eindruck, um sich der generischen Problematik anzunehmen. Jedoch haben wir auch gemerkt, dass ein festes Kontingent an Modellierungskomponenten nicht ausreicht, um möglichst verschiedene und komplexe Systeme zu simulieren. Somit muss neben den vorgegebenen Komponenten die Möglichkeit gegeben sein, eigene Komponenten zu implementieren. Da im SimTech Cluster nicht jeder Forscher Programmierkenntnisse besitzt, muss neben textueller Codeimplementierung, eine weitere Interaktionsmethode gegeben sein. Diese soll Forschern ohne Programmierkenntnisse ermöglichen, eigene Komponenten zu implementieren. Als besonders empfehlenswert empfinden wir die Methode Programmieren durch Demonstrieren (PBD), in Kombination mit visuellen Vorher-Nachher Regeln. Dadurch lassen sich annähernd gleichwertige Komponenten implementieren, wie sie durch herkömmlichen Code entstehen würden.

6. Zusammenfassung

Für den SimTech Cluster der Universität Stuttgart soll ein Simulationswerkzeug, das möglichst wenig Programmiererfahrung voraussetzt, entwickelt werden. Zur Bestandsaufnahme und Informationsbeschaffung wurde in dieser Fachstudie eine Umfrage erstellt und auf Grund der Ergebnisse eine Empfehlung für mögliche Herangehensweisen an ein solches Werkzeug gegeben. Angefangen wurde mit der Untersuchung von verwandten Arbeiten. Dabei wurde der Fokus auf allgemeine Informationen zum Thema Simulation, sowie auf Forschungsergebnissen im Bereich der Simulationswerkzeuge für Programmieranfänger, gelegt. Um einen Überblick über genutzte Simulationswerkzeuge im SimTech Cluster der Universität Stuttgart zu erlangen, wurde eine Umfrage erstellt. Ziel dieser Umfrage war nicht allein die Bestandsaufnahme verwendeter Programme, sondern auch die Beschaffung von zusätzlichen Informationen über Interaktionsmöglichkeiten und nötige Programmiererfahrung. Die Auswertung der Umfrage und der darin genannten Simulationswerkzeuge zeigte, dass Programme meist sehr speziell für einen Fachbereich entwickelt sind. Dabei werden oft hohe Ansprüche an die Programmierkenntnisse des Nutzers gestellt. Eine Programm mit zusammenfassenden Funktionen ist nur schwer zu realisieren.

Auf Grund dieser Auswertung haben wir eine Empfehlung ausgesprochen, die sich auf Interaktionsmöglichkeiten für das Werkzeug fokussiert. Um für Programmieranfänger geeignet zu sein, muss ein Werkzeug einen generischen Ansatz verfolgen. Dabei macht die Blockdiagrammnotation zur Modellierung Sinn. Außerdem muss das Programm erweiterbar sein, um auch speziellere Vorgehensweisen zu unterstützen. Da dies selten ohne Programmierkenntnisse realisierbar ist, empfehlen wir die Methode Programmieren durch Demonstrieren (PBD), in Kombination mit visuellen Vorher-Nachher Regeln.

7. Danksagung

An dieser Stelle wollen wir uns bei allen Beteiligten an der Fachstudie bedanken.

Besonderer Dank geht an unsere Betreuer Frau Greis und Herrn Lischke, für hilfreiche Tipps und Hilfestellungen während der gesamten Studie.

Außerdem wollen wir uns bei allen Teilnehmern der Onlineumfrage bedanken, die eine Aussagekräftige Empfehlung erst ermöglicht haben.

Besonderer Dank gilt auch den Mitarbeitern des SimTech Clusters, die uns zu einem Interview zur Verfügung standen. Dabei sind nicht nur die vorangegangen Interviews mit hilfreichen Tipps zur Umfrage gemeint, sondern auch die abschließenden Interviews zu tieferen Einblicken in die Arbeit.

A. Anhang

A.1. Fragebogen

A.1.1. Generelle Fragen

1. Your Name: (Textfeld)
2. Your EmailAddress: (Textfeld)
3. Your Occupation:
    a) PhD Student
    b) Post-doctorand
    c) Professor
    d) Other: (Textfeld)
4. Your highest degree:
    a) Bachelor
    b) Master
    c) Diploma
    d) PhD
    e) Habituation
    f) Other: (Textfeld)
5. Your project network:
    a) PN1: Material Simulation
    b) PN2: High Performance Simulation
    c) PN3: Dynamic Systems
    d) PN4: Biomechanics & Systembiology
    e) PN5: Multiphase- and Multiphysics Modelling
    f) PN6: Cyber-Infrastructure
    g) PN7: Reflection & Contextualization

   h) Not a member of SimTech

6. Which university, institute or company are you working for? (Textfeld, aber nur für den Fall falls die darüberliegende Frage mit h) beantwortet wurde, sonst wurde diese Frage nicht angezeigt)
7. Your Subject:
   a) Mathematics
   b) Computer science
   c) Engineering
   d) Social sciences
   e) Economics
   f) Sciences
   g) Other: (Textfeld)

A.1.2. Modellierung- und Simulationsprozess

1. On which step(s) of the process are you currently working on? (Mehrere Antworten möglich)
   a) Data gathering
   b) Modelling
   c) Simulation
   d) Visualization
   e) Interpretation
   f) Other: (Textfeld)
2. What are you specifically doing in this/these step(s)? (Textfeld)
3. What modelling/simulation approach are you using? (Textfeld)
4. Please classify your approach: (Mehrere Antworten möglich)
   a) Static
   b) Dynamic
   c) Continous
   d) Discrete
   e) Stochastic
   f) Deterministic
   g) Other: (Textfeld)

5. Which modelling/simulation programs do you know best in your working field? (Textfeld)
6. Do you use self-written programs?
    a) Yes
    b) No
7. Which programs are you using for your work? (Textfeld)
8. Please rate your general programming skills: (Von 1 bis 5, der Fähigkeit nach aufsteigend)

A.1.3. Programm Bewertung

1. Name of the program: (Textfeld)
2. In which step(s) of the process do you use this program? (Mehrere Antworten möglich)
    a) Data gathering
    b) Modelling
    c) Simulation
    d) Visualization
    e) Interpretation
    f) Other: (Textfeld)
3. How are you primarily using this program?
    a) Graphical interface
    b) Command line
    c) Program-specific code
    d) External code
    e) Other: (Textfeld)
4. Please indicate your level of agreement with the following statements: (Bewertung von eins bis fünf, nach aufsteigender Zustimmung)
    a) It is simple to use this system.
    b) I can effectively complete my work using this system.
    c) I am able to complete my work quickly using this system.
    d) I am able to efficiently complete my work using this system.
    e) It was easy to learn to use this system. strongly disagree
    f) The system gives error messages that clearly tell me how to fix problems.

g) Whenever I make a mistake using the system, I recover easily and quickly.

h) The information (such as online help, on-screen messages, and other documentation) provided with this system is clear.

i) It is easy to find the information I need.

j) The information provided with the system is easy to understand.

k) I like using the interface of this system.

l) Overall, I am satisfied with this system. strongly disagree

5. Does the program need any improvement?

   a) Yes

   b) No

6. What kind of improvement is needed? (Textfeld - Nur wenn die vorherige Frage mit Ja beantwortet wurde)

7. For how many years are you using this program now?

   a) 0-1

   b) 1-2

   c) 2-5

   d) 5+

8. How much of your working time are you using this program (roughly)? (Prozentuale Angabe)

9. Do you need a programming language to opperate the program?

   a) Yes

   b) No

10. Which one? (Textfeld - Nur wenn die vorherige Frage mit Ja beantwortet wurde)

11. Please indicate your level of agreement with the following statements: Bewertung von eins bis fünf, nach aufsteigender Zustimmung)

    a) I am an expert user of this program.

    b) Good programming skills are required to use this program.

    c) I have good programming skills in the needed language(s).

12. Does the program offer an integrated editor for program-specific code?

    a) Yes

    b) No

1. Is it okay that the marked answers are shared with other SimTech employees?

   a) Yes
   b) No
2. Would you like to be informed about the results of the survey?
   a) Yes
   b) No
3. Would you be willing to provide us insight into your work?
   a) Yes
   b) No
4. Please feel free to leave any comments: (Textfeld)

A.2. Ergebnisse

A.2.1. Details der Schritte

- -Visualization and communication of untercertainty -Development of a simulation tool for non-experts
- Adaptive sampling, GPU computing, Parameter studies
- Compare Results for different kinds of models/ Benchmark tests
- The research is on the development of a simulation method that is able to couple discrete and continuous methods.
- Modelling and simulating molecules
- implementing a parallel 2dgrid
- Use standard methods for simulating fluid motions, implement and check whether they are sufficient. Improve them to get better results.
- deriving models; Validation with measurement results
- - plot data - compare with literature - evaluation of experimental data
- data mining model developement parameter estimation
- Analysing the role of societal uncertainty and complexity in simulations
- Modelling the system based on Newton's Laws by using various integrator schemes, performing numerical simulations and calculating statics and dynamical properties of the system.
- Multiscale simulation
- Code development and supervision of PhD students

- Deriving and preparation of model equations numerical solving
- 1.) Set up a set of competing conceptual models 2.) express each as mathematical model 3.) Set up a set of prior knowledge (probability distribution) for model parameters and model credibilities 4.) Perform forward uncertainty quantification 5.) perform inverse uncertainty quantification and probabilistic risk assessment 6.) use the final (set of) models for decision support 7.) analyze the impact of possible future data collection on the improvement of the decision problem
- - development of efficient discretization for partial differential equations - fast and scalable solvers for systems of linear equations - numerical methods for coupling black-box solvers to a multiphysics simulation environment - efficient parallel implemenations of dynamically adaptive computational grids
- produce different models and compare them against each other
- Coupling of fluid-structure-acoustics solver.
- Modelling: stochastic description of input parameters for simulations Interpretation: estimating stochastic properties of some quantities of interest for some simulations
- Implementation of new numerical algorithms
- Modelling: deciding which atoms to take into our model and which to neglect, which approximations to make when solving Schrödinger's equation, ... Visualization: Viewing of potential energy surfaces, reaction paths, ...
- Contextualizing simulations at the science-policy interface
- I am gathering data in order to model a simulation model about investment decisions and I try to create a management flight simulator to measure the impact of the simulation model on human decsion making
- Modelling Car Side Impact Scenarios including Dummy/Human Modells in LSDyna. Modelling muscles in a simple leg modell in NewEulM2.
- Skeletal muscle modelling
- Conducting expert interviews for gathering qualitative data about the society - energy system interface.
- Modelling aggregation process of nanoparticles
- Change parameters, run simulation, visualize results, interpret results extend constitutive equation
- parameter estimation biochemical modeling uncertainty analysis statistical learning methods
- perform simulations
- making sense of large social data bases
- identifying parameters representing biological terms, implementing numerically the model approach

- Uncertainty Quantification
- method development, coupling atomistic and continuum models
- Trying to find a mathematical description and transfer that to computer code.
- Creating models to explain physical processes happening on the atomistic scale, implementing them in existing academic code, and predicting experimental observations.
- Implementing algorithms and simulating the results as well as the performance and the bahvior of the modelled system w.r.t. this algorithm.
- Data gathering: Collect data using Participatory Sensing: Users carry mobile devices sensing the environment Simulation: Developing mechanisms to run simulations on mobile devices leveraging resources of stationary servers connected over the internet.
- I'm designing feedback controllers for PDEs and compare them to lower order controllers, which are cheaper to calculate.
- Modelling of a chemical Production Network
- Development of numerical methods and computer programs
- simulate muscle movement and activation
- Literature review and trying to develop an understanding of the modelling framework in general.

A.2.2. Im Bereich bekannte Werkzeuge

- Matlab (15)
- (Open)CMISS (5)
- DUNE (4)
- ESPResSo (4)
- C++(4)
- Python (3)
- DUMUX(2)
- ParaView (2)
- GROMACS (2)
- Maple (2)
- preCICE (2)
- OpenFOAM (2)
- COMSOL (2)
- LAMMPS (2)
- Eclipse (2)
- NetLogo
- Stagecast
- StarLogo
- Excel
- Fortran
- C
- mathematica
- dymola
- amesim

- demoa
- OpenSim
- Anybody
- GID
- Ovito
- VMD
- FRESHS
- Mobile (c++ multi-body dynamics engine)
- 3Dcreate
- MODFLOW
- Xpert-N, Hydrogeosphere
- ANSYS Fluent
- Alya (Barcelona Supercomputing Center)
- Peano (inhouse code)
- DAKOTA (Sandia)
- UQTk (Sandia)
- UQLab (ETH)
- ChemShell
- DL-FIND
- Molpro
- Turbomole
- Vensim
- iThink
- Stella
- PowerSim
- Anylogic
- Consideo iModeller
- LSDyna
- Hypermesh
- Hypercrash
- LSPrepost
- NewEulM2
- ScenarioWizard
- graphviz
- siesta
- gromacs
- vasp
- namd
- SAS
- PANDAS
- SG++
- IMD
- PLATO
- EDAMAME
- ONETEP
- GAUSSIAN
- ORCA
- Simulink
- GNU R
- Tau-Code
- CFX
- FEBio
- OpenSIM

A.2.3. Für die Arbeit verwendete Programme

- Matlab (13)
- DUMUX (4)
- OpenCMISS (4)
- C++ (4)
- ESPResSo (3)
- DUNE (3)
- SG++ (3)
- Exel(2)
- power-point (2)
- Linux (2)
- LaTeX (2)
- Eclipse (2)
- NumPro (2)
- Python (2)
- Maple (2)
- Ovito (2)
- Simulink (2)
- git(2)
- COMSOL(2)
- LS-DYNA (2)
- Sublime Text
- SVN
- Word
- NetLogo
- StarLogo
- Fortan
- ParaView
- GCC
- LLVM
- mathematica
- dymola
- amesim
- demoa
- GID
- Paraview
- VMD
- FRESHS
- Visual Studio 2010
- MODFLOW
- Xpert-N
- Hydrogeosphere
- ANSYS Fluent
- Alya (Barcelona Supercomputing Center)
- Peano (inhouse code)
- preCICE (inhouse code)
- OpenFOAM
- SBToolbox
- preCICE
- Shell
- Emacs
- KMail
- LAMMPS
- Visit
- ChemShell
- DL-FIND
- Molpro
- Turbomole
- Vensim
- Hypermesh
- Hypercrash

- LSPrepost
- NewEulM2
- ABAQUS
- ScenarioWizard
- OpenCell
- siesta
- gromacs
- SAS
- SPPS
- MABSS
- PANDAS
- PLATO
- FLEXI
- EDAMAME

A.2.4. Programmberwertung

C++,Eclipse

Wird in Prozessschritten verwendet: Modellierung, Simulieren
Verwendet wird es über ein graphisches UI.

Es ist einfach das System zu benutzen.	Stimmt
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Stimmt
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Stimmt
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.1.: CSUQ-Eclipse,C++

Keine Verbesserungen benötigt
Es wird seit zwei bis fünf Jahren verwendet und das 80% der Arbeitszeit.
Zur Verwendung wird C++ benötigt. Der Teilnehmer sagt, er kann das Werkzeug mittelmäßig benutzen, wobei Sehr gute C++ Kenntnisse benötigt sind, die der Teilnehmer auch hat. Das Werkzeug liefert einen integrierten Editor mit.

ChemShell

Wird in Prozessschritten verwendet: Simulieren
Verwendet wird es Durch Programm-spezifischen Code.

Es ist einfach das System zu benutzen.	Stimmt
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt sehr
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt sehr
Es war einfach das System zu erlernen.	Stimmt
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Neutral
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt
Es ist einfach benötigte Informationen zu finden.	Stimmt sehr
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Stimmt sehr
Generell bin ich mit dem System zufrieden.	Stimmt sehr

Tabelle A.2.: CSUQ-ChemShell

Gewünschte Verbesserungen sind bessere Simulationsprotokolle und effizientere Algorithmen
Es wird seit mehr als fünf Jahren verwendet und das 10% der Arbeitszeit.
Zur Verwendung wird keine Programmiersprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug sehr gut benutzen. Das Werkzeug liefert keinen integrierten Editor mit.

demoa

Wird in Prozessschritten verwendet: Modellierung, Simulieren
Verwendet wird es Durch programm- spezifischen Code.

Es ist einfach das System zu benutzen.	Stimmt weniger
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Neutral
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Neutral
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt weniger
Es ist einfach benötigte Informationen zu finden.	Neutral
Informationen die vom System kommen sind einfach zu verstehen.	Neutral
Ich benutze gerne das Interface des Systems.	Neutral
Generell bin ich mit dem System zufrieden.	Stimmt überhaupt nicht

Tabelle A.3.: CSUQ-demoa

Gewünschte Verbesserungen sind weitere Entwicklung und Dokumentation
Es wird seit zwei bis fünf Jahren verwendet und das 75% der Arbeitszeit.
Zur Verwendung wird C bzw. C++ als Programmiersprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug sehr gut benutzen. Um das Programm zu bedienen werden gute Programmiererfahrungen vorausgesetzt diese hat der Teilnehmer auch. Das Werkzeug liefert keinen integrierten Editor mit.

EMACS

Wird in Prozessschritten verwendet: Simulieren
Verwendet wird es über ein graphisches UI.

Es ist einfach das System zu benutzen.	Stimmt weniger
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Neutral
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt sehr
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Neutral
Ich benutze gerne das Interface des Systems.	Stimmt weniger
Generell bin ich mit dem System zufrieden.	Stimmt sehr

Tabelle A.4.: CSUQ-EMACS

Keine Verbesserungen benötigt
Es wird seit mehr als fünf Jahren verwendet und das 75% der Arbeitszeit.
Zur verwendung wird Lisp benötigt. Der Teilnehmer sagt, er kann das Werkzeug gut benutzen, wobei durchschnittliche Lisp Kenntnisse benötigt sind, die der Teilnehmer nicht ausreichend hat. Das Werkzeug liefert einen integrierten Editor mit.

GAMS

Wird in Prozessschritten verwendet: Modellierung
Verwendet wird es über eine Kommandozeile.

Es ist einfach das System zu benutzen.	Stimmt sehr
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt sehr
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt sehr
Es war einfach das System zu erlernen.	Stimmt
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt weniger
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt weniger
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt
Es ist einfach benötigte Informationen zu finden.	Stimmt weniger
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Stimmt
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.5.: CSUQ-GAMS

Keine Verbesserungen benötigt
Es wird seit bis zu einem Jahr verwendet und das 20% der Arbeitszeit.
Zur Verwendung wird Python benötigt. Der Teilnehmer sagt, er kann das Werkzeug recht gut benutzen, wobei durchschnittliche Python Kenntnisse benötigt sind, in denen der Teilnehmer sehr gute hat. Das Werkzeug liefert einen integrierten Editor mit.

GROMACS

Wird in Prozessschritten verwendet: Modellierung, Simulieren
Über die Verwendung und Bedienung wurden keine Angaben gemacht.

HyperCrash

Wird in Prozessschritten verwendet: Modellierung
Verwendet wird es über ein graphisches UI.

Es ist einfach das System zu benutzen.	Stimmt sehr
Ich kann meine Arbeit effektiv mit dem System erledigen.	Neutral
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Neutral
Es war einfach das System zu erlernen.	Stimmt sehr
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt weniger
Es ist einfach benötigte Informationen zu finden.	Stimmt weniger
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt weniger
Ich benutze gerne das Interface des Systems.	Stimmt
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.6.: CSUQ-HyperCrash

Es ist eine Verbesserung gewünscht insofern, dass das ausgaben manchmal falsch und nicht strukturiert sind.
Es wird seit bis zu einem Jahr verwendet und das 30% der Arbeitszeit.
Zur Verwendung wird keine Programmiersprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug gut benutzen. Das Werkzeug liefert keinen integrierten Editor mit.

LS-DYNA

Wird in Prozessschritten verwendet: Simulieren
Verwendet wird es über eine Kommandozeile.

Es ist einfach das System zu benutzen.	Neutral
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Stimmt weniger
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt überhaupt nicht
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt weniger
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Neutral
Es ist einfach benötigte Informationen zu finden.	Neutral
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt weniger
Ich benutze gerne das Interface des Systems.	Stimmt weniger
Generell bin ich mit dem System zufrieden.	Neutral

Tabelle A.7.: CSUQ-LS-DYNA

Keine Verbesserungen benötigt
Es wird seit bis zu einem Jahr verwendet und das 10% der Arbeitszeit.
Zur Verwendung wird keine Programmiersprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug gut benutzen. Das Werkzeug liefert keinen integrierten Editor mit.

LS-PREPOST

Wird in Prozessschritten verwendet: Modellierung, Visualisierung
Verwendet wird es über ein graphisches UI.

Es ist einfach das System zu benutzen.	Neutral
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Stimmt weniger
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt überhaupt nicht
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt überhaupt nicht
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt weniger
Es ist einfach benötigte Informationen zu finden.	Stimmt weniger
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt weniger
Ich benutze gerne das Interface des Systems.	Neutral
Generell bin ich mit dem System zufrieden.	Neutral

Tabelle A.8.: CSUQ-LS-PREPOST

Interface variiert stark zwischen unterschiedlichen Versionen und somit sind Guides nur sehr spezifisch auf die eine Interface version anzuwenden
Es wird seit bis zu einem Jahr verwendet und das 10% der Arbeitszeit.
Zur Verwendung wird keine Programmiersprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug benutzen (Neutral). Das Werkzeug liefert einen integrierten Editor mit.

Maple

Wird in Prozessschritten verwendet: Modellierung, Simulieren
Verwendet wird es über eine Kommandozeile.

Es ist einfach das System zu benutzen.	Neutral
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Stimmt weniger
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Neutral
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Neutral
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Neutral
Es ist einfach benötigte Informationen zu finden.	Stimmt weniger
Informationen die vom System kommen sind einfach zu verstehen.	Neutral
Ich benutze gerne das Interface des Systems.	Stimmt
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.9.: CSUQ-Maple

Keine Verbesserungen benötigt
Es wird seit mehr als fünf Jahren verwendet und das 10% der Arbeitszeit.
Zur Verwendung wird Programm-spezifischer Code benötigt. Der Teilnehmer sagt, er kann das Werkzeug benutzen, wobei geringe Kenntnisse über den Programm-spezifischer Code benötigt sind, über die der Teilnehmer aber mittelmäßig beherrscht. Das Werkzeug liefert einen integrierten Editor mit.

Molpro

Wird in Prozessschritten verwendet: Modellierung
Verwendet wird es über Programm-spezifischer code.

Es ist einfach das System zu benutzen.	Neutral
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt sehr
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt sehr
Es war einfach das System zu erlernen.	Stimmt weniger
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt weniger
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Neutral
Es ist einfach benötigte Informationen zu finden.	Neutral
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Stimmt sehr
Generell bin ich mit dem System zufrieden.	Stimmt sehr

Tabelle A.10.: CSUQ-Molpro

Keine Verbesserungen benötigt
Es wird seit mehr als fünf Jahren verwendet und das 10% der Arbeitszeit.
Zur Verwendung wird keine Sprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug sehr gut benutzen. Das Werkzeug liefert keinen integrierten Editor mit.

NetLogo

Wird in Prozessschritten verwendet: Modellierung, Simulieren, Visualisierung, Interpretation
Verwendet wird es über Programm-spezifischer Code.

Es ist einfach das System zu benutzen.Stimmt	
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Stimmt
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt sehr
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Neutral
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt sehr
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Stimmt
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.11.: CSUQ-NetLogo

Keine Verbesserungen benötigt
Es wird seit bis zu einem Jahren verwendet und das 5% der Arbeitszeit.
Zur Verwendung wird Netlogo als Sprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug mittelmäßig benutzen, wobei gute Netlogo Kenntnisse benötigt sind, die der Teilnehmer auch hat.
Das Werkzeug liefert einen integrierten Editor mit.

NumPro

Wird in Prozessschritten verwendet: Simulieren
Verwendet wird es über Kommandozeile.

Es ist einfach das System zu benutzen.	Neutral
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Neutral
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt weniger
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Neutral
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Neutral
Es ist einfach benötigte Informationen zu finden.	Neutral
Informationen die vom System kommen sind einfach zu verstehen.	Neutral
Ich benutze gerne das Interface des Systems.	Neutral
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.12.: CSUQ-NumPro

Gewünschte Verbesserungen sind, weitere Entwicklung, Dokumentation und einen code der es ermöglicht diesen auch Kommerziell zu verwenden.
Es wird seit mindestens fünf Jahren verwendet und das 50% der Arbeitszeit.
Zur Verwendung wird C++ benötigt. Der Teilnehmer sagt, er kann das Werkzeug sehr gut benutzen, mittelmäßige C++ Kenntnisse benötigt sind, die der Teilnehmer auch hat. Das Werkzeug liefert einen integrierten Editor mit.

Pandas

Wird in Prozessschritten verwendet: Simulieren
Verwendet wird es über externen Code.

Es ist einfach das System zu benutzen.	Stimmt weniger
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt sehr
Es war einfach das System zu erlernen.	Stimmt weniger
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Neutral
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Neutral
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt weniger
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Neutral
Ich benutze gerne das Interface des Systems.	Neutral
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.13.: CSUQ-Pandas

Gewünschte Verbesserungen sind, ein Interface und Unterstützung von parallel computing
Es wird seit bis zu einem Jahre verwendet und das 40% der Arbeitszeit.
Zur Verwendung wird C benötigt. Der Teilnehmer sagt, er kann das Werkzeug gar nicht benutzen, wobei sehr gute C Kenntnisse benötigt sind, die der Teilnehmer aber nur mittelmäßig hat. Das Werkzeug liefert keinen integrierten Editor mit.

ParaView

Wird in Prozessschritten verwendet: Visualisierung
Verwendet wird es über ein graphisches UI.

Es ist einfach das System zu benutzen.	Neutral
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt überhaupt nicht
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Stimmt
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt weniger
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Neutral
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt weniger
Es ist einfach benötigte Informationen zu finden.	Stimmt weniger
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Stimmt weniger
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.14.: CSUQ-ParaView

Als Verbesserung wurde ein Neuladen-Button vorgeschlagen, um besser mit inkorrekten Visualisierungs-daten zu arbeiten.
Es wird seit zwei bis fünf Jahren verwendet und das 10% der Arbeitszeit.
Zur Verwendung wird keine Programmiersprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug eher weniger benutzen. Das Werkzeug liefert keinen integrierten Editor mit.

Plato

Wird in Prozessschritten verwendet: Modellierung, Simulieren
Verwendet wird es über eine Kommandozeile

Es ist einfach das System zu benutzen.	Stimmt weniger
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt sehr
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt sehr
Es war einfach das System zu erlernen.	Stimmt
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt überhaupt nicht
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt überhaupt nicht
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt weniger
Es ist einfach benötigte Informationen zu finden.	Stimmt weniger
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt weniger
Ich benutze gerne das Interface des Systems.	Stimmt weniger
Generell bin ich mit dem System zufrieden.	Stimmt sehr

Tabelle A.15.: CSUQ-Plato

Gewünschte Verbesserungen sind eine bessere Modalisierung, effizientere Daten Strukturen und eine bessere Bedienbarkeit
Es wird seit bis zu einem Jahr verwendet und das 70% der Arbeitszeit.
Zur Verwendung wird C benötigt. Der Teilnehmer sagt, er kann das Werkzeug sehr gut benutzen, wobei gute C Kenntnisse benötigt sind, die der Teilnehmer auch hat. Das Werkzeug liefert keinen integrierten Editor mit.

SAS

Wird in Prozessschritten verwendet: Modellierung, Simulieren
Verwendet wird es über Programm-spezifischen Code.

Es ist einfach das System zu benutzen.	Stimmt weniger
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt weniger
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Stimmt weniger
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt überhaupt nicht
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt weniger
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt weniger
Es ist einfach benötigte Informationen zu finden.	Neutral
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt weniger
Ich benutze gerne das Interface des Systems.	Neutral
Generell bin ich mit dem System zufrieden.	Stimmt weniger

Tabelle A.16.: CSUQ-SAS

Ein besseres Nutzerinterface ist erwünscht.
Es wird seit mehr als fünf Jahren verwendet und das 10% der Arbeitszeit.
Zur Verwendung wird keine Programmiersprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug weniger gut benutzen, Das Werkzeug liefert keinen integrierten Editor mit.

Scenario Wizard

Wird in Prozessschritten verwendet: Daten sammeln, Modellierung, Simulieren, Visualisierung, Interpretation
Verwendet wird es über ein graphisches UI.

Es ist einfach das System zu benutzen.	Stimmt
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt sehr
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt sehr
Es war einfach das System zu erlernen.	Neutral
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Neutral
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Neutral
Ich benutze gerne das Interface des Systems.	Stimmt sehr
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.17.: CSUQ-Scenario Wizard

Gewünschte Verbesserungen sind, mehr Postprozess Unterstützung, eine Assistentenfunktion für Anfänger und mehr Analysebeispiele in der Dokumentation
Es wird seit mehr als fünf Jahren verwendet und das 20% der Arbeitszeit.
Zur Verwendung wird keine Programmiersprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug sehr gut benutzen. Das Werkzeug liefert keinen integrierten Editor mit.

Self-written Java for Android

Wird in Prozessschritten verwendet: Daten Sammeln
Verwendet wird es über externen Code.

Es ist einfach das System zu benutzen.	Stimmt
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt sehr
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt sehr
Es war einfach das System zu erlernen.	Stimmt
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Neutral
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt weniger
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Neutral
Ich benutze gerne das Interface des Systems.	Stimmt weniger
Generell bin ich mit dem System zufrieden.	Neutral

Tabelle A.18.: CSUQ-Self-written Java for Android

Keine Verbesserungen benötigt
Es wird seit ein bis zwei Jahren verwendet und das 2% der Arbeitszeit.
Zur Verwendung wird Java benötigt. Der Teilnehmer sagt, er kann das Werkzeug gut benutzen, wobei gute Java Kenntnisse benötigt sind, die der Teilnehmer auch hat. Das Werkzeug liefert keinen integrierten Editor mit.

Self-written Python

Wird in Prozessschritten verwendet: Daten Sammeln, Simulieren
Verwendet wird es über Programm-spezifischen Code.

Es ist einfach das System zu benutzen.	Stimmt
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Stimmt
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Neutral
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Stimmt
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.19.: CSUQ-Self-written Python

Keine Verbesserungen benötigt
Es wird seit zwei bis fünf Jahren verwendet und das 10% der Arbeitszeit.
Zur Verwendung wird Python benötigt. Der Teilnehmer sagt, er kann das Werkzeug gut benutzen, wobei gute Python Kenntnisse benötigt sind, die der Teilnehmer auch hat. Das Werkzeug liefert keinen integrierten Editor mit.

SG++

Wird in Prozessschritten verwendet: Modellierung, Interpretation
Verwendet wird es als Bibliothek.

Es ist einfach das System zu benutzen.	Stimmt sehr
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt sehr
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt sehr
Es war einfach das System zu erlernen.	Neutral
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt weniger
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt weniger
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Stimmt
Generell bin ich mit dem System zufrieden.	Stimmt sehr

Tabelle A.20.: CSUQ-SG++

Als Verbesserung ist gewünscht, dass auch andere Sprachen als C++ unterstützt werden
Es wird seit zwei bis fünf Jahren verwendet und das 80% der Arbeitszeit.
Zur Verwendung wird ist C++ am angenehmsten, aber Python, Java und Matlab sind auch unterstützt.
Der Teilnehmer sagt, er kann das Werkzeug gut benutzen, wobei gute C++ Kenntnisse benötigt sind, die der Teilnehmer auch hat. Das Werkzeug liefert keinen integrierten Editor mit.

Siesta

Wird in Prozessschritten verwendet: Modellierung, Simulieren
Verwendet wird es über eine Kommandozeile.

Es ist einfach das System zu benutzen.	Stimmt
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Stimmt
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Neural
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Neutral
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Neutral
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Stimmt überhaupt nicht
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.21.: CSUQ-Siesta

Keine Verbesserungen benötigt
Es wird seit mehr als fünf Jahren verwendet und das 50% der Arbeitszeit.
Zur Verwendung wird keine Programmiersprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug mittelmäßig benutzen. Das Werkzeug liefert keinen integrierten Editor mit.

Vensim

Wird in Prozessschritten verwendet: Modellierung, Simulieren, Visualisierung
Verwendet wird es über ein graphisches UI.

Es ist einfach das System zu benutzen.	Stimmt
Ich kann meine Arbeit effektiv mit dem System erledigen.	Neutral
Ich kann mit dem System meine Arbeit schnell erledigen	Neutral
Ich kann meine Arbeit effizient mit dem System erledigen.	Neutral
Es war einfach das System zu erlernen.	Stimmt weniger
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt überhaupt nicht
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Neutral
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt sehr
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Stimmt
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.22.: CSUQ-Vensim

Es ist ein mehr intuitives Interface gewünscht, da das Werkzeug viele Funktionen hat, die dem Teilnehmer unbekannt sind
Es wird seit zwei bis fünf Jahren verwendet und das 15% der Arbeitszeit.
Zur Verwendung wird keine Programmiersprache benötigt. Der Teilnehmer sagt, er kann das Werkzeug mittelmäßig benutzen. Das Werkzeug liefert einen integrierten Editor mit.

Details CSUQ

Es ist einfach das System zu benutzen.	Stimmt
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Stimmt
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Stimmt
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.23.: CSUQ-MATLAB

Es ist einfach das System zu benutzen.	Stimmt überhaupt nicht
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt sehr
Ich kann mit dem System meine Arbeit schnell erledigen	Neutral
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Stimmt weniger
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Neutral
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Neutral
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Neutral
Es ist einfach benötigte Informationen zu finden.	Neutral
Informationen die vom System kommen sind einfach zu verstehen.	Neutral
Ich benutze gerne das Interface des Systems.	Neutral
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.24.: CSUQ-DUNE

Es ist einfach das System zu benutzen.	Neutral
Ich kann meine Arbeit effektiv mit dem System erledigen.	Neutral
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt weniger bis Neutral
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt weniger bis Neutral
Es war einfach das System zu erlernen.	Stimmt weniger bis überhaupt nicht
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt weniger bis Neutral
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Neutral
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Neutral
Es ist einfach benötigte Informationen zu finden.	Neutral
Informationen die vom System kommen sind einfach zu verstehen.	Neutral
Ich benutze gerne das Interface des Systems.	Stimmt
Generell bin ich mit dem System zufrieden.	Stimmt

Tabelle A.25.: CSUQ-(Open)CMISS

Es ist einfach das System zu benutzen.	Neutral
Ich kann meine Arbeit effektiv mit dem System erledigen.	Stimmt
Ich kann mit dem System meine Arbeit schnell erledigen	Stimmt
Ich kann meine Arbeit effizient mit dem System erledigen.	Stimmt
Es war einfach das System zu erlernen.	Neutral
Das System gibt deutliche Fehlermeldungen, die mir klar sagen wie Probleme zu lösen sind.	Stimmt
Sobald ich einen Fehler beim benutzen des Systems mache, kann ich schnell und einfach Aktionen rückgängig machen	Stimmt
.Hilfestellungen zum System (z.B. Online-Hilfe, Systemmeldungen und andere Dokumentation) sind deutlich.	Stimmt
Es ist einfach benötigte Informationen zu finden.	Stimmt
Informationen die vom System kommen sind einfach zu verstehen.	Stimmt
Ich benutze gerne das Interface des Systems.	Neutral
Generell bin ich mit dem System zufrieden.	Stimmt (sehr)

Tabelle A.26.: CSUQ-ESPResSo

Literaturverzeichnis

[Bro99] J. F. Broenink. *20-SIM software for hierarchical bond-graph/block-diagram models.* 1999. (Zitiert auf den Seiten 18 und 19)

[DCST00] A. C. David Canfield Smith, L. Tesler. *Novice Programming Comes of Age.* 2000. (Zitiert auf den Seiten 6, 9, 19, 21 und 22)

[Gora] G. Gordon. *The Development of the General Purpose Simulation System (GPSS).* (Zitiert auf den Seiten 9, 14, 15, 16, 17 und 19)

[Gorb] G. Gordon. *A General Purpose Systems Simulation Program.* (Zitiert auf den Seiten 9, 14, 15, 16 und 19)

[KP05] C. Kelleher, R. Pausch. *Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers.* 2005. (Zitiert auf den Seiten 9, 19, 20, 21 und 22)

[LKK91] A. M. Law, W. D. Kelton, W. D. Kelton. *Simulation modeling and analysis*, Band 2. McGraw-Hill New York, 1991. (Zitiert auf den Seiten 12 und 13)

[MI99] J. C. Martinez, P. G. Ioannou. *General-Purpose Systems for Effective Construction Simulation.* 1999. (Zitiert auf den Seiten 14, 18 und 19)

[SB11] J. A. Sokolowski, C. M. Banks. *Principles of modeling and simulation: a multidisciplinary approach.* John Wiley & Sons, 2011. (Zitiert auf den Seiten 9, 11, 12, 13 und 14)

[SB12] J. A. Sokolowski, C. M. Banks. *Handbook of Real-world Applications in Modeling and Simulation*, Band 2. John Wiley & Sons, 2012. (Zitiert auf Seite 12)

[Sol86] E. Soloway. *Learning to Program = Learning to Construct Mechanisms and Explanations.* 1986. (Zitiert auf den Seiten 19 und 21)

[TLR] K. S. Tim Laue, T. Röfer. *SimRobot - A General Physical Robot Simulator and its Application in RoboCup.* (Zitiert auf Seite 19)

[VK11] A. O. M. H. P. G. E. T. Vahur Kotkas, Riina Maigre. *CoCoViLa as a Multifunctional Simulation Platform.* 2011. (Zitiert auf den Seiten 17, 18 und 19)

Alle URLs wurden zuletzt am 28. 09. 2014 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift