Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminarausarbeitung

Parham Vasaiely

Offene Simulation und Test von Systemen mit SysML, Modelica und Eclipse

Faculty of Engineering and Computer Science
Department Computer Science

Fakultät Techchnik und Informatik
Department Informatik

Parham Vasaiely

Parham.Vasaiely@haw-hamburg.de

Offene Simulation und Test von Systemen mit SysML, Modelica und Eclipse

Ausarbeitung eingereicht im Rahmen der Veranstaltung Seminar

im Studiengang Informatik (Master)
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuer:
Prof. Dr.-Ing. Bettina Buth
Prüfer:
Prof. Dr. rer. nat. Kai von Luck
Prof. Dr. rer. nat. Gunter Klemke

Abgegeben am 22. Februar 2011

Parham Vasaiely

Thema des Projekts

Offene Simulation und Test von Systemen mit SysML Modelica und Eclipse

Stichworte

UML, SysML, Modelica, Simulation, interaktive Simulation, System, modell-basierte Entwicklung, Systementwicklung, Test, Testautomatisierung, Anforderungen, Eclipse

Kurzzusammenfassung

In dieser Ausarbeitung werden die Arbeiten an dem Masterprojekt mit dem Thema „Systemmodellierung, Test und Simulation mit UML/SysML und Modelica“ aufgeführt. Insgesamt wurden drei Hauptthemen des Masterprojekts, Modellierung, Simulation und Test von technischen Systemen, aufgegriffen und im Rahmen einer ersten Analyse und prototypischen Entwicklung behandelt. Nachfolgend werden erste Lösungen zu einer grafischen Modellierungssprache, einer modifizierten und erweiterten Simulationslaufzeitumgebung, einer Simulationsumgebung basierend auf der Eclipsetechnologie und einem geeigneten dynamischen Testverfahren aufgeführt, deren Entwicklung und die dadurch gewonnen Resultate erläutert.

Title of the project

Open System Simulation and Test with SysML, Modelica and Eclipse

Keywords

UML, SysML, Modelica, Simulation, interactive simulation, System, Model based Engineering,

Systems Engineering, Test, Test automation, Requirement, Eclipse

Abstract

This paper is a report of the research performed and applications developed during a master project with the title “System Modeling, Simulation and Testing with UML/SysML and Modelica”. The master project covers three main topics, the modeling, simulation and testing of technical systems. In this paper all three parts, the modeling, the simulation and the testing part have been managed, analyzed and a first prototype has been developed. In this paper you can find a possible graphical modeling language, a brief description of performed modification of an existing simulation runtime environment to offer an interactive simulation with Modelica and a simulation environment based on the Eclipse 3.6 plug-in technology. Also a process of dynamic testing has been reviewed.

Inhaltsverzeichnis

I. Abbildungsverzeichnis

II. Glossar

CAE	Computer Aided Engineering
INCOSE	International Council on Systems Engineering
MBSE	Model-Based Systems Engineering
OM	OpenModelica
OMC	OpenModelica Compiler
OMG	Object Management Group
OMI	OpenModelica Interactive
OPENPROD	Open Model-Driven Whole-Product Development and Simulation Environment
SUT	System Under Test
SysML	Systems Modelling Language

1. Einleitung

In dieser Ausarbeitung sollen in kurzer Form die wichtigsten Motivationsgründe und das Ziel dieser Arbeit erläutert werden. Ebenfalls werden die gewonnen Erkenntnisse, des im Sommersemester 2010 stattgefundenen „Projekt 1“ und des im Wintersemester 2010/11 stattgefundenen „Projekt 2“, aufgeführt. Ein Masterprojekt, welches im günstigsten Fall in einer Masterarbeit resultiert, wird in mehrere Teilaufgaben bzw. Teilprojekte aufgegliedert. Das „Seminar“ dient der Rekapitulation gewonnener Erkenntnisse und soll eine konkrete und fortgeschrittene Planung der daraus resultierenden Masterarbeit darstellen. Diese beinhaltet neben den zu behandelnden Themen auch eine Risikoanalyse.

1.1. Hintergrund und Motivation

Dieses Projekt findet im Rahmen des Open Model-Driven Whole-Product Development and Simulation Environment (OPENPROD) Projektes statt, welches ein von der Europäischen Union als ITEA2-Projekt [4], gefördertes und von führenden Industrieunternehmen, Forschungsinstituten und Universitäten durchgeführtes Forschungsprojekt ist. Das OPENPROD wurde angeregt durch das International Council on Systems Engineering (INCOSE) [6] welches die modellbasierte Systementwicklung (MBSE) [6] als eine wichtige Schlüsseltechnik zur effizienten und effektiven Entwicklung von Systemen in der Zukunft identifizierte [7]. Eine standardisierte Notation zur Beschreibung der Systemanforderungen oder des Systemdesigns an jedem beliebigen Punkt der Entwicklungsphase, ist eine der wichtigsten Techniken in der MBSE. Ebenfalls hat sich die Simulation von technischen Systemen, beispielsweise zu Analyse- und Validierungszwecken, in der Praxis als ein großer Vorteil des MBSE herausgestellt [8].

Während den Recherchen des OPENPROD-Teams zum „Stand der Wissenschaft und Technik“ im Bereich der Systementwicklung, wurden UML und SysML als zwei standardisierte grafische Modellierungssprachen [9][10], Modelica als Simulationsfähige Modellierungssprache [3], OpenModelica als Simulationslaufzeitumgebung [11] und Eclipse als freiverfügbare und nicht kommerzielle Entwicklungsumgebung [13] identifiziert [14].

Die Entwicklung eines durchgängigen und frei verfügbaren Verfahrens zur Förderung der MBSE und Unterstützung der Systementwickler stellt somit eine große Herausforderung dar und bietet genügend Themen für interessante, praxisrelevante wissenschaftliche Arbeiten [14]. Mehr Informationen zu diesem Thema finden Sie in der Ausarbeitung [15].

1.2. Ziele dieser Arbeit

Die aus dem Semesterübergreifenden Masterprojekt resultierende Arbeit fokussiert auf Probleme und Aufgabenstellungen, welche bei der Entwicklung technischer Systeme entstehen:

- Standardisierte (Grafische) Modellierung der Systeme und deren Teilkomponenten, unter Verwendung von UML/SysML [9][10] und Modelica [3].
- Analyse, Validierung des Systems und der Teilkomponenten durch Simulation derselben.
- Validierung und Unterstützung bzw. Verbesserung der Qualitätssicherung durch dynamisches Testen des Systems und dessen Teilkomponenten.

Die erarbeiteten Grundlagen gilt es durch die technische Umsetzung in Form von prototypischen SW-Werkzeugen zugänglich zu machen. Die technischen Ziele lassen sich wie folgt zusammenfassen:

- Erarbeitung und Erweiterung offener standardisierter Beschreibungssprachen (Domänen- und Unternehmensübergreifend) und Zwischenformate wie z.B. XML zur Transformation eines UML/SysML Modells in simulationsfähigen Modelica Code
- Integration der Methoden und Verfahren in freiverfügbare Engineering- und Simulationswerkzeuge.
- Nachweis der Einsetzbarkeit der Methoden und Werkzeuge anhand von Demonstratoren.

Weitere Ziele dieser Arbeit finden Sie in der Ausarbeitungen[15].

1.3. Ziele dieser Ausarbeitung

Im Rahmen des im Wintersemester 2010/11 stattgefundenen „Seminars“ sind folgende Punkte behandelt worden. Wobei entweder eine erste, aber dennoch konkrete, Analyse der Problemstellung mit Lösungsansätzen oder ersten prototypischen Entwicklungen bzw. Implementierungen angestrebt wurde.

- Ermittlung einer geeigneten grafischen Modellierungssprache basierend auf UML/SysML und Modelica 2.
- Modifikation und Erweiterung der frei verfügbaren Simulationslaufzeitumgebung OpenModelica zur Simulation von Modelica Modellen 3.
- Ermittlung und Analyse von geeigneten dynamischen Testverfahren zum automatischen oder interaktiven Test von Systemen und deren Komponenten 4.
- Risikoanalyse und die Bewertung derselben nach der Wahrscheinlichkeit ihres Auftretens und die damit verbundenen Auswirkungen auf das Gesamtprojekt.

2. Systemmodellierung

In diesem Kapitel soll die Notwendigkeit eines mathematischen Modells zur Simulation und Test von Systemen erläutert und die zu verwendende Modellierungssprache zur Repräsentation desselben aufgeführt werden.

Ein Modell ist das vereinfachte und auf das Wesentliche, für die entsprechende Entwicklungsphase, abstrahierte Abbild eines realen Systems oder einer Teilkomponente. Modelle werden zur Veranschaulichung, Kommunikation, Interaktion und Analyse verwendet. Eine Modellierungssprache hilft dem Entwickler die Anforderungen an ein System und dessen Struktur in einer abstrakteren Ebene darzustellen [6][8].

2.1. Mathematische Modelle

Um ein technisches System noch vor seiner physikalischen Entwicklung, mit Hilfe eines Rechners simulieren und testen zu können wird die Abbildung auf ein virtuelles, mathematisches Modell benötigt. Ein mathematisches Modell drückt die Beziehung zwischen Variablen in mathematischer Form aus [3]. Die Variablen bilden dabei die Eigenschaften des Systems ab, wie zum Beispiel die Größe, Masse oder Geschwindigkeit. In dieser Arbeit wird Modelica [3] zur Repräsentation der mathematischen Modelle verwendet. Modelica ist eine frei verfügbare, objektorientierte Sprache für die Modellierung von großen, komplexen und heterogenen physikalischen Systemen. Modelle in Modelica werden durch Differentialgleichungen und diskrete Gleichungen mathematisch beschrieben. Keine Variable muss von Hand gelöst werden, da ein Modelica- Werkzeug genügend Informationen besitzt, dies automatisch zu entscheiden.

```
block Model_XY
    ...
equation
    der(x) = error / T;
    outCtr = K * (error + x);
end Model_XY;
```

Abbildung 2-1 Modelica Code-Beispiel für eine mathematische Gleichung

2.2. Grafische Modellierungssprache

Im Bereich der Softwareentwicklung haben sich mit der UML als grafische Modellierungssprache und den vielfältigen Programmiersprachen offene Standards bezüglich Spezifikation, Entwurf und Implementierung etabliert [1][2]. Modelica und SysML zeigen großes Potential, diese Lücke auch im Bereich der Systementwicklung zu schließen [16][4]. So mal SysML und Modelica sich ergänzende Sprachen für die Systementwicklung sind. SysML ist eine Beschreibungssprache für

Systemmodelle und Modelica eine formale, gleichungsbasierte Modellierungssprache [18]. In Kapitel 3 der Arbeit [16] wurde das reine SysML 1.1 zur Modellierung verwendet und anschließend in Modelica Code transformiert. Dieser Ansatz ist nur unter Einschränkungen nutzbar, da die Elemente der beiden Modellierungssprachen nicht eins-zu-eins aufeinander abgebildet werden können. Nachfolgend werden zwei Projekte vorgestellt, welche eine deutlich weiterführende Integration von UML/SysML und Modelica erreichen möchten und im Rahmen des „Masterprojektes“ analysiert und bewertet wurden:

2.2.1. SysML and Modelica Integration

Unter der Führung von führenden OMG SysML Entwicklern wie Chris Paredis und Sandy Friedenthal sowieso dem Modelica Entwickler Peter Fritzson, wurde im Dezember 2008 eine Arbeitsgruppe gegründet welche die Integration von SysML und Modelica untersuchen und bei positivem Ausgang einen Transformationsansatz für SysML und Modelica entwickeln soll [29]. Das entstandene UML2 Profil nennt sich „SysML4Modelica“, alle verwendeten Stereotypen und Erweiterungen zu SysML können in der Arbeit [30] unter „Part II“ nach verfolgt werden.

2.2.2. ModelicaML

Aus der Arbeitsgruppe zur „SysML und Modelica Integration“ entstand eine eigene Arbeitsgruppe welche nicht das reine SysML 1.1 sondern UML2 als Erweiterungsgrundlage nutzt. Mit ModelicaML [31] wurde daher ein UML-Profil zur Integration der beiden Notationen vorgeschlagen, welches aber auch die Systemspezifischen Vorteile von SysML respektiert. ModelicaML bietet dem Benutzer die Möglichkeit UML2 Struktur- und Verhaltensdiagramme bei der Entwicklung seines Modells zu verwenden.

3. Systemsimulation

„Simulation ist die Nachbildung eines dynamischen Prozesses in einem „realen“ Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind“ [27]. Der Erkenntnisgewinn wird durch Experimente erreicht, welche auf einem mathematischen Modell durchgeführt werden und maßgeblich zur Validierung der dynamischen Teile eines entwickelten Modells beitragen. Mehr zum Einsatz von Simulationen und die verschiedenen Simulationsarten finden Sie in Kapitel 2.2 [15].

3.1. Simulationslaufzeitumgebung OpenModelica

Um eine Simulation der entwickelten Modelica Modelle zu gewährleisten ist neben einer Simulationsumgebung auch eine Simulationslaufzeitumgebung nötig [16]. Diese Laufzeitumgebung macht die Modelle ausführbar und gewährleistet die korrekte Simulation durch Verwendung eines Gleichungslösers (engl. Solver). Neben einer Reihe kommerzieller Tools [28] bietet OpenModelica (OM), als einziges quelloffenes und nicht kommerzielles Werkzeug, die Möglichkeit Modelica Modelle zu Kompilieren und den generierten Code in eine Simulationslaufzeit zu integrieren [12]. Daher wurde es schon in der Projektbeschreibung [15] und im ITEA2 als zu verwendendes Werkzeug gefordert [14].

3.1.1. OpenModelica Interactive

OpenModelica bietet dem Benutzer eine nicht-interaktive und somit asynchron zur Echtzeit ablaufende Simulationsart. Da dem Benutzer die Möglichkeit gegeben werden soll, sein Systemmodell prototypisch, interaktiv und synchron zur Echtzeit zu simulieren, muss OM um einige Fähigkeiten und Komponenten erweitert werden, die dies möglich machen.

Das im „Projekt 1“ entstandene und nachfolgend erläuterte Modul wurde von den OpenModelica Entwicklern als eine geeignete Erweiterung von OM anerkannt und wurde daher, während eines Entwicklertreffens an der Linköping University, in das offizielle OM Projekt übernommen. Dieses Modul wird als „OpenModelica Interactive“ (OMI) bezeichnet und ist seit der OpenModelica Version 1.6.0 Teil des Simulationslaufzeitkerns [11]. Um OM um die interaktiven Komponenten zu erweitern wird der als Simulationslaufzeit existierende Code in Subsysteme aufgeteilt. Dieses Prinzip unterstützt die Modularisierung und das „Information Hiding“. Die Aufteilung in Subsysteme ist an die serviceorientierte Architektur angelehnt, welches Vorteile bei dem Austausch, der Modifikation und Erweiterung von einzelnen Systemteilen bietet [23].

Folgende zwei Subsysteme wurden identifiziert:

- Die existierenden, modifizierten oder erweiterten Teile, als OM- Subsystem.
- Die neuen interaktiven Komponenten, als OMI- Subsystem.

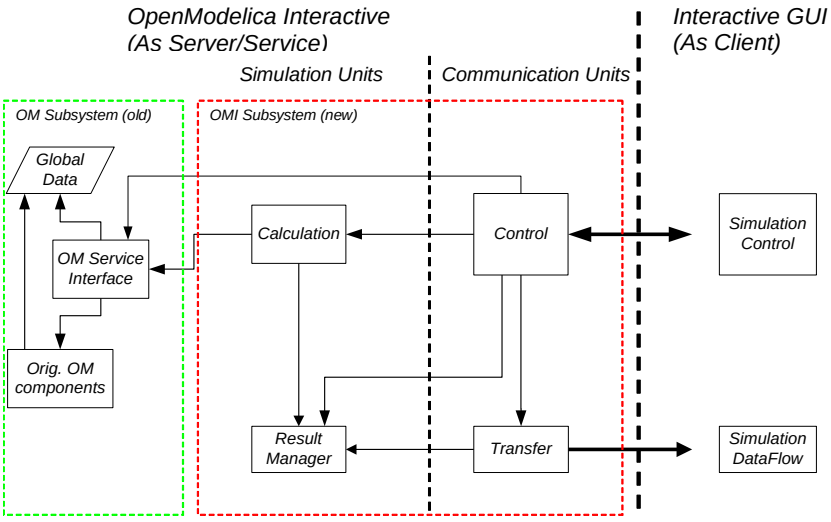

Abbildung 3-1 Überblick der Simulationslaufzeit Subsysteme und ihrer Komponenten

OpenModelica Subsystem:

- Ursprüngliche OpenModelica Komponenten. Das Modul wurde um ein Service-Interface erweitert welches die Kontrolleinheit beim Zugriff auf die OM Komponenten aus anderen Subsystemen darstellt.

OpenModelica Interactive Subsystem:

- OMI::Control: Diese Modul dient als Schnittstelle zum Befehlsaustausch zwischen der Simulationslaufzeit und einer Simulationsumgebung. Basierend auf einem Netzwerk Kommunikationsprotokoll [Kapitel 5.1.3 [16]] können Befehle und Informationen ausgetauscht werden.
- OMI::ResultManager: Verwaltung und Organisation von Simulationsergebnissen
- OMI::Calculation: Verantwortlich für den Zeitsynchronen Aufruf des Gleichungslöser aus dem OM-Subsystem.
- OMI::Transfer: Verantwortlich für die Zeitsynchrone Bereitstellung von Simulationsergebnissen für eine quasi in Echtzeit ablaufende interaktive Simulation.

3.2. Simulationsumgebung

Die Simulationsumgebung ermöglicht dem Benutzer die nicht-interaktive sowie die interaktive Simulation eines in UML2 oder SysML erstellten Systems. Die Schnittstelle zu OM wird zur Erzeugung einer Simulationslaufzeit verwendet, welche als lauffähige Applikation das System und Gleichungslöser (Solver) zur Simulation beinhaltet. Ebenfalls wird eine Interaktionsschnittstelle

zwischen Simulation und Benutzer, zur Kontrolle einer interaktiven Simulation in Form von Start, Stopp und Pause Befehlen sowieso die visuelle Änderung von Systemparametern, gewährleistet. Des Weiteren bietet das Tool dem Benutzer eine einfache visuelle Darstellung von Ergebnissen als Graphen.

3.2.1. Simulation Plug-in

Als eine nichtkommerzielle und quelloffene Entwicklungsumgebung bietet die Eclipsetechnologie dem Benutzer durch seinen modularen Aufbau die Möglichkeit eigene Erweiterungen als sogenannte „Eclipse Plug-Ins“ zu entwickeln und in die Umgebung zu integrieren. Die resultierende Erweiterung kann zusammen mit dem Eclipse Kern als eigenständige Applikation in Rich Client Platform (RCP) [24] oder als ein nicht eigenständiger Zusatz in Plug-in Development Environment (PDE) [25] entwickelt werden. Die hauptsächliche Entwicklung basiert auf der Programmiersprache Java.
Nachfolgend werden einige der im „Projekt 1“ entwickelten Plug-Ins aufgeführt und erläutert.

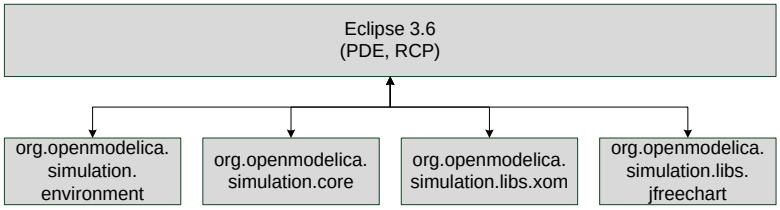

Abbildung 3-2: Übersicht der Eclipse Plug-ins für die Simulationsumgebung

org.openmodelica.simulation.core: Dieses Plug-In stellt alle benötigten Komponenten zur Simulation eines Modelica Modells in Java bereit. Seine Komponenten sind auf die Verwendung mit OpenModelica beschränkt, aber unabhängig von einer speziell implementierten Simulationsumgebung einsetzbar.
org.openmodelica.simulation.libs.xom: Zur Initialisierung von Parametern, Konfiguration von Simulationswerten werden XML Daten eingesetzt. XOM ist eine nicht kommerzielle Bibliothek zum Lesen und Schreiben von XML Files. Aktuell wird die Version 1.2.4 verwendet [21].
org.openmodelica.simulation.libs.jfreechart: Zur Darstellung der als Zahlen vorliegenden Simulationsergebnisse kommt JFreeChart zum Einsatz. Dies ist eine sehr ausgereifte und für den nicht kommerziellen Bereich, frei zugängliche, Bibliothek zur Darstellung von mathematischen Diagrammen. Zum Einsatz kommt die Version 1.0.13 [20].
org.openmodelica.simulation.environment: Dies ist die konkrete Implementierung einer Simulationsumgebung als Plug-In für Eclipse. Die Anforderungen an diese Simulationsumgebung wurden in Zusammenarbeit mit Systementwicklern bei der EADS Innovation Works Hamburg erstellt.

4. Systemtest

Das Testen ist eines der wichtigsten Methoden zur strukturellen Analyse und anschließenden Auswertung der Ergebnisse eines Modells hinsichtlich seiner statischen und dynamischen Struktur. Ein Systemtest ist ein Test, welcher während der Systementwicklung durchgeführt wird. Die Notwendigkeit von Tests ist durch folgende Begründungen gegeben:

- Fehlerwirkungen frühzeitig erkennen um ihre Fehlerzustände zu beseitigen.
- Komponententest und deren Integrationstest als Gesamtsystem zur Qualitätssicherung.
- Verifikation und Validierung des Systems und der Einzelkomponenten.

4.1. Funktionale Tests

Die funktionale Spezifikation eines Systems beschreibt das Verhalten, welches von einem System eingehalten werden muss, um die vom Kunden gewünschten Ergebnisse zu liefern. Dessen Umsetzung ist Voraussetzung dafür, dass das System überhaupt einsetzbar ist [32]. Die Einhaltung dieser Anforderungen wird mit Hilfe von funktionalen Tests überprüft.

4.2. Dynamischer Test

Anders als beim statischen wird beim dynamischen Test ein System zur Ausführung gebracht [32]. Im Kontext einer Simulation wird das System-Under-Test (SUT) ausgeführt indem sein, als mathematisches Modell repräsentiertes, Verhalten auf Softwareprozesse abgebildet wird.

4.2.1. Black-box Verfahren

Bei dem Black-Box Testverfahren werden die Testfälle basierend auf der funktionalen Spezifikation erstellt. Im Fokus stehen dabei hauptsächlich die erzeugten Ausgaben eines SUT, unter Berücksichtigung definierter Eingaben und Bedingungen. Bewertet werden die Daten anhand ermittelter Ist- sowie Sollwerte. Die interne Struktur eines Systems wird, anders als beim White-Box Test [x] nicht betrachtet.

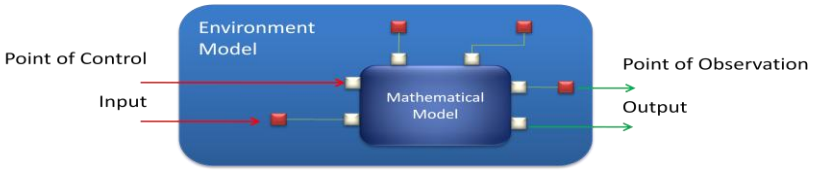

Abbildung 4-1 Black-Box Test

Beim Black-Box Test liegt der „Point of Control“ (PoC), eine Schnittstelle, über die das Testobjekt mit Testdaten versorgt wird, und der „Point of Observation“ (PoO), die Schnittstelle, an der die Reaktionen und Ausgaben des Testobjekts beobachtet wird, außerhalb des zu testenden Modells [32].

4.3. Testrealisierung

Um die Durchführung vorzubereiten müssen Testfälle nach der Generierung Bewertet, Geordnet und Priorisiert werden [32]. Die Testrealisierung umfasst hierbei u.a. folgende Schritte:

- Konkretisieren und Priorisieren der Testfälle
- Erstellung von Testdaten und Testszenarien
- Zusammenstellen von Testmengen und Testsuiten
- Ordnung der Testsuiten in ein Testskript

4.4. Testdurchführung

Nachfolgend sind zwei Methoden zur Durchführung von Testfällen nach der Realisierung aufgeführt.

4.4.1. Automatische Testdurchführung

Um den Entwickler oder Tester bestmöglich und unkompliziert zu unterstützen, sollte die Testdurchführung weitgehend automatisch, durch ein entsprechendes Testwerkzeug, ausgeführt werden. Hierbei kommen die während der Testrealisierung erstellten Testskripte zum Einsatz.

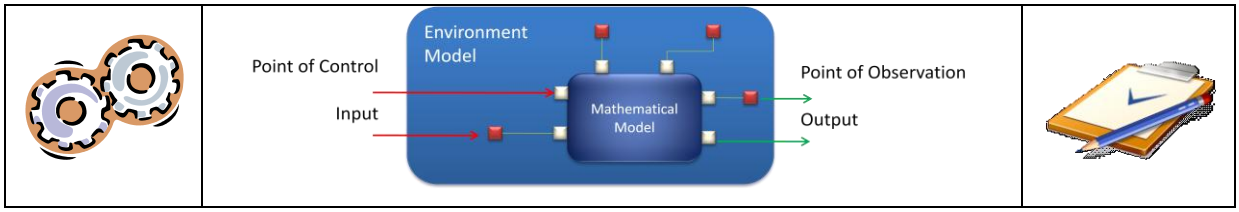

Abbildung 4-2 Automatische Testdurchführung

4.4.2. Interaktive Testdurchführung

Sehr oft kommt es vor, dass die funktionale Spezifikation eines Systems unvollständig ist oder nicht alle vom Anwender gewünschten Eigenschaften an dasselbe enthält [33]. Da allerdings bei der Testautomatisierung Testfälle mit Hilfe dieser Spezifikation abgeleitet werden, kann eine interaktive bzw. manuelle Testdurchführung, bei der ein Anwender das Systemmodell interaktive stimuliert, weitere kritische Fehlerwirkungen aufdecken.

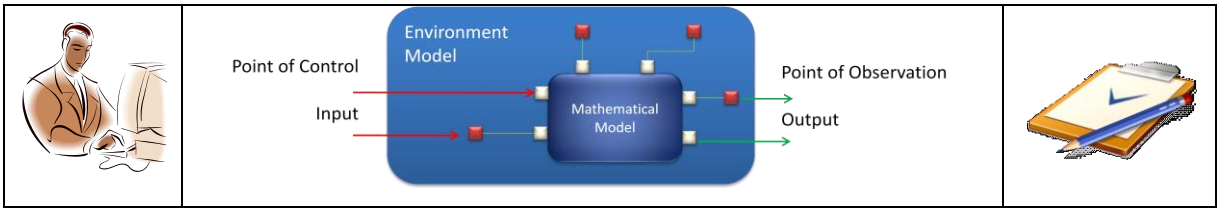

Abbildung 4-3 Interaktive Testdurchführung

5. Risiken

Dieses Kapitel soll einige Risiken und deren Einschätzung aufführen, welche im Rahmen einer ersten Risikoanalyse das Ergebnis dieser Arbeit negative beeinflussen würden.

Da OpenModelica ein Universitätsprojekt ist, entstehen laufend neue Arbeiten zu diesem Thema. Eine daraus entstehende parallele Entwicklung an dem Quellcode könnte zu Fehlersituation führen. Da dieses Problem in der Vergangenheit bereits aufgetreten ist und einen nicht unerheblichen Zeitaufwand zur Lösung mit sich gezogen hat, kann ein solches Auftreten mit einer hohen Risikostufe bewertet werden.

Die eingeschränkte Verwendbarkeit von SysML zur Modellierung von Modelica Modellen und anschließenden automatischen Testfallgenerierung könnte sich als Problem darstellen. Da sich die Verwendung allerdings auf ein klar definiertes Beispiel beschränkt, können komplizierte Implementierungen von vornherein, durch weitere Einschränkungen am System, verhindert werden. Des Weiteren kann immer noch auf die UML Sprachelemente zurückgegriffen werden. Dieses Risiko ist daher niedrig einzustufen.

Ein ebenfalls nicht zu unterschätzendes Thema dieser Arbeit ist der Systemtest. Das automatische Testen umfasst die Testfallgenerierung, Realisierung und Durchführung. In Verbindung mit dem model-basierten Testen könnte dies zu unerwarteten Schwierigkeiten führen, da die Einschätzung der Komplexität im vornherein schwer ist. Da mit Frau Prof. Dr. Buth eine erfahrene Testspezialistin, als Projektbetreuerin gewonnen werden konnte, kann dieses Risiko auf mittel gestuft werden.

6. Zusammenfassung

Die modelbasierte Systementwicklung mit Unterstützung von UML/SysML und Modelica zeigt großes Potenzial die Probleme während der Systementwicklung durch einheitliche und freizugängliche Methoden und Applikationen zu lösen.

Die unter 2.2.1 und 2.2.2 aufgeführten Ansätze zur Integration von UML/SysML und Modelica bieten eine deutlich höhere Abdeckung der beiden Sprachen, sind allerdings nur im Rahmen der Forschung einsetzbar und keine Standardsprachen wie das reine OMG SysML es ist. Da diese Arbeit ebenfalls nur die Simulation und den Test fokussiert, reichen die in der Arbeit [16] gewonnen Ergebnisse aus, um die nötige Transformation von SysML zu Modelica und die damit erstellten mathematischen Modelle im Rahmen der Forschung und einer ersten prototypischen Entwicklung einsetzen zu können.

Die Entwicklung der unter 3.1.1 aufgeführten Erweiterung von OpenModelica und die damit verbundene Einbindung in das offizielle OpenModelica Projekt, zeigen das diese Arbeit durchaus den Anspruch einer über die Forschung hinaus gehenden Arbeit mit Praxisrelevanz hat [12].

Die Testautomatisierung umfasst neben der automatischen Testfallgenerierung auch die automatische Testrealisierung und Durchführung. Bei der Durchführung können allerdings Fehlerwirkungen verborgen bleiben, falls die zu ihrer Erkennung benötigten Testfälle nicht aus der Spezifikation hergeleitet werden können. Hier kann die interaktive bzw. manuelle Testdurchführung Abhilfe schaffen.

III. Referenzen

[1] Sanford Friedenthal, Alan Moore and Rick Steiner, 2008, Practical Guide to SysML: The Systems Modeling Language, Morgan Kaufmann.

[2] Tim Weilkiens, 2008, Systems Engineering mit SysML/UML, dpunkt.Verlag

[3] Fritzson Peter, 2004, Principles of Object-Oriented Modeling and Simulation with Modelica 2.1, Wiley-IEEE Press.

[4] ITEA 2 Project, Last Accessed: 2010, http://www.itea2.org/

[5] The International Council on Systems Engineering (INCOSE), Last Accessed: 2010 http://www.incose.org/

[6] INCOSE Model Based Systems Engineering (MBSE), Last Accessed: 2010 http://mbse.gfse.de/

[7] Friedenthal, Sanford, Greigo, Regina, and Mark Sampson, INCOSE MBSE Roadmap, in "INCOSE Model Based Systems Engineering (MBSE) Workshop Outbrief" (Presentation Slides), presented at INCOSE International Workshop 2008, Albuquerque, NM, pg. 6, Jan. 26, 2008.

[8] SysML Project: Telescope systems modelling by SE^2, Last Accessed: 2010 http://mbse.gfse.de/documents/32.html

[9] Object Management Group UML, "OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2", http://www.omg.org/docs/formal/07-11-02.pdf, November 2007.

[10] Object Management Group SysML, "OMG Systems Modeling Language (OMG SysML™) Specification", http://www.omg.org/docs/formal/08-11-02.pdf, November 2008.

[11] OpenModelica Project, Last Accessed: 2010, http://openmodelica.org/

[12] PELAB, Peter Fritzson, "OpenModelica System Documentation, Version, 2010-07-06 for OpenModelica1.5.0", http://www.ida.liu.se/labs/pelab/modelica/OpenModelica/releases/1.5.0/doc/OpenModelicaSystem.pdf, July 2010.

[13] Eclipse Platform and Eclipsetechnologie, Last Accessed: 2010 www.eclipse.org

[14] Open Model-Driven Whole-Product Development and Simulation Environment (OPENPROD) Full Project Proposal, (Internes Dokument), 2009-11-09

[15] Parham Vasaiely, “Ausarbeitung - Anwendungen 1”, http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/vasaiely/bericht.pdf, December 2009

[16] EADS Innovation Works, HAW Hamburg, Parham Vasaiely, Bachelor Thesis: “Interactive Simulation of SysML Models using Modelica.pdf”, August 2009. http://opus.haw-hamburg.de/volltexte/2009/842/

[17] Computing and Math. Research Division Lawrence Livermore National Laboratory, Petzold, Linda R., DDASSL Fortran source code: http://www.netlib.org/ode/ddassl.f, December 2006.

[18] Modelica and the Modelica Association, Modelica Libraries, Last Accessed: 2009 http://www.modelica.org/libraries

[19] Open Source Modelica Consortium, Last Accessed: 2009 http://www.ida.liu.se/labs/pelab/modelica/OpenSourceModelicaConsortium.html

[20] JFreeChart Eclipse Plug-In, Java chart library, Last Accessed: 2010 http://www.jfree.org/jfreechart

[21] XOM Eclipse Plug-In, API for processing XML, Last Accessed: 2010 http://www.xom.nu

[22] Andrew S. Tanenbaum, 2003, Computer Networks (4th Edition), Prentice Hall.

[23] Ralf Reussner und Wilhelm Hasselbring, 2006, Handbuch der Software-Architektur, dpunkt.

[24] Eclipse Rich Client Platform, Last Accessed 2010 http://www.eclipse.org/rcp

[25] Eclipse Plug-In Development Environment, Last Accessed 2010 http://www.eclipse.org/pde

[26] MathCore, MathModelica Lite with OpenModelica kernel, Last Accessed: 2010 http://www.mathcore.com/products/mathmodelica/lite/

[27] Verein Deutscher Ingenieure, Simulation von Logistik-, Materialfluss- und Produktionssystemen, http://www.vdi.de/uploads/tx_vdirili/pdf/1398802.pdf, October 2009

[28] DASSAULT SYSTEMES, Dymola, Last Accessed: 2010, www.dymola.com

[29] Object Management Group (OMG), SysML and Modelica Integration, Last Accessed: 2010, http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-modelica:sysml_and_modelica_integration

[30] Object Management Group (OMG) SysML and Modelica Integration Specification, http://www.omgwiki.org/OMGSysML/lib/exe/fetch.php?id=sysml-modelica%3Asysml_and_modelica_integration&cache=cache&media=sysml-modelica:sysml-modelica_xformspec_v.1.0_2010-5-10.pdf, May 2010

[31] Wladimir Schamai, Modelica Modeling Language (ModelicaML), Last Accessed: 2010 http://openmodelica.org/index.php/component/content/article/34-main-category/134-modelica-modeling-language-modelicaml

[32] Andreas Spillner, Tilo Linz, 2010, Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester: Foundation Level nach ISTQB-Standard, dpunkt.verlag GmbH

[33] Christof Ebert, 2008, Systematisches Requirements Engineering und Management, dpunkt.verlag GmbH