Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Ausarbeitung - Anwendungen 2

Parham Vasaiely

Related Work zum Thema:
System Modellierung, Test und Simulation
mit UML/SysML und Modelica

August 2010

Faculty of Engineering and Computer Science
Department Computer Science

Fakultät Techchnik und Informatik
Studiendepartment Informatik

**Parham Vasaiely**

**Thema des Projekts**

System Modellierung, Test und Simulation mit UML/SysML und Modelica

**Stichworte**

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

**Kurzzusammenfassung**

In dieser Arbeit werden verwandte Projekte vorgestellt die, im Rahmen einer ersten Analyse als hilfreich und zweckmäßig, zur Lösung von Teilproblemen einer Methode und Anwendung zur modellbasierten Systementwicklung mit UML/SysML und Modelica identifiziert wurden. Neben der Modellierung mit einer standardisierten grafischen Modellierungssprache und einer freizugänglichen Modellierungsumgebung, wurde ebenfalls das Problem der Kommunikation und des Datenaustausches während einer laufenden nichtinteraktiven oder interaktiven Simulation behandelt.

**Title of the project**

System modelling, testing and simulation with UML/SysML and Modelica

**Keywords**

UML, SysML, Modelica, Simulation, interactive-Simulation, System, Model based Engineering, Systems Engineering, Test, Test automation, Requirement, Eclipse

**Abstract**

This paper presents four approaches which are related to the “System modelling, testing and simulation with UML/SysML and Modelica” project. Each approach focuses on a sub problem of developing a method and application of model-based systems engineering with UML/SysML and Modelica. In addition to the modeling with a standardized modeling language and a free available modeling environment, the communication and data transfer problem while simulating a system non-interactively or interactively will also be respected.

# Inhaltsverzeichnis

# I. Abbildungsverzeichnis

# II. Glossar

| | |
|---|---|
| CAE | Computer Aided Engineering |
| GML | Graphical Modelling Language |
| INCOSE | International Council on Systems Engineering |
| MBSE | Model-Based Systems Engineering |
| MDA | Model Driven Architektur |
| MOF | Meta Object Facilities |
| OM | OpenModelica |
| OMC | OpenModelica Compiler |
| OMG | Object Management Group |
| OMI | OpenModelica Interactive |
| QVT | Query View Transformation |
| SysML | Systems Modelling Language |
| U2TP | UML 2.0 Test Profile |
| UML | Unified Modelling Language |

# 1. Einleitung

In dieser Ausarbeitung sollen in kurzer Form der Hintergrund und die wichtigsten Motivationsgründe zu dieser Arbeit erläutert werden. Ebenfalls werden verwandte Projekte vorgestellt, die im Rahmen einer ersten Analyse als hilfreich und zweckmäßig zur Lösung von Teilproblemen des Projektes identifiziert wurden.

## *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 Industrie Unternehmen, Forschungs- Instituten und Universitäten angefragtes Forschungsprojekt ist. Das OPENPROD wurde angeregt durch das International Council on Systems Engineering (INCOSE) [5] 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, wurde die UML/SysML als standardisierte grafische Modellierungssprache [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 somit genügend Themen für interessante, praxisrelevante wissenschaftliche Arbeiten [14]. Mehr Informationen zu diesem Thema und den Zielen dieses Projektes finden Sie in der Ausarbeitungen [15].

# 2. Related Work

Die hier aufgeführten Projekte werden zur Lösung von drei Teilbereichen in Betracht gezogen, welche jeweils Problemstellungen dieser Arbeit darstellen.
Die folgenden Teilbereiche sollen hier behandelt werden:

- Grafische Modellierungssprache: Eine standardisierte grafische Modellierungssprache (GML) wird benötigt um Systemmodelle visuell darstellen zu können.
- Grafische Modellierungsumgebung: Unter Verwendung einer grafischen Modellierungsumgebung (Modelling plug-in) kann, mit Einsatz einer gewählten GML, ein System modelliert und seine Funktionen und Eigenschaften visualisiert werden.
- Simulation von Systemen: Die Simulation von dynamischen Systemteilen wird zu Analyse- und Validierungszwecken genutzt.

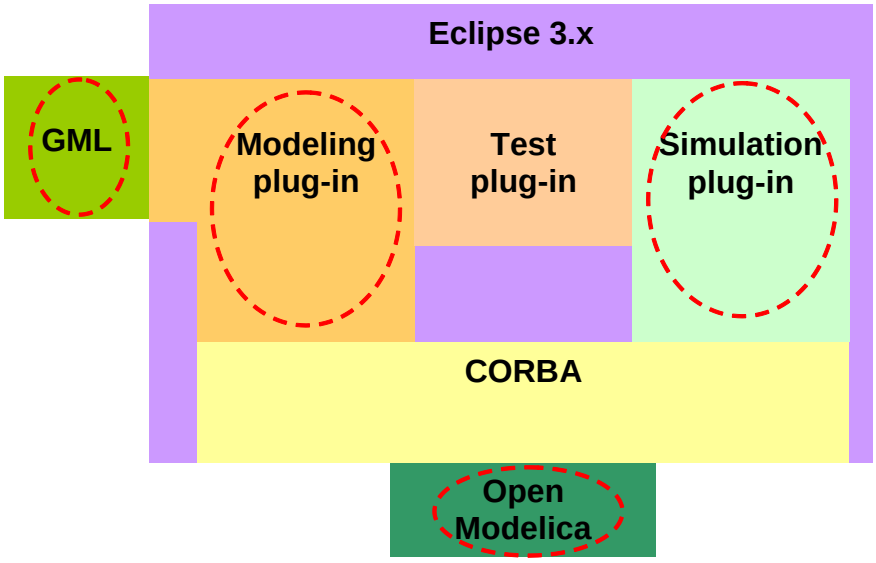


**Abbildung 2-1 Darstellung der in dieser Arbeit behandelten Teilbereiche**

## *2.1. 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]. Im 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.
Da Modelica als Programmiersprache nicht ohne weiteres modifiziert/erweitert werden kann, liegt es nahe die Erweiterungsfunktionen von UML/SysML zu nutzen um diese Sprache an Modelica anzupassen, beispielsweise durch Stereotypen [1].
Nachfolgend werden zwei Projekte vorgestellt die einmal SysML und einmal UML als Grundlage ihrer Erweiterung (UML- Profil) verwenden.

### 2.1.1. SysML und 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 [17].
SysML und Modelica sind sich ergänzende Sprachen für die Systementwicklung. SysML ist eine Beschreibungssprache für Systemmodelle und Modelica eine formale, gleichungsbasierte Modellierungssprache [18].
Um zwei Sprachen transformieren zu können wird ein „bi-direktionales mapping“, also die Abbildung von Sprachelemente aufeinander, benötigt. Diese Abbildung kann allerdings nur entstehen, wenn ein gemeinsamer Kontext zwischen beiden Sprachen besteht. Hierbei muss zuerst ein zu Modelica kompatibles Profil von SysML und die abstakte Syntax von Modelica, ein Metamodell [18] definiert und erzeugt werden.
Das entstandene Profil nennt sich „SysML4Modelica“, alle verwendeten Stereotypen und Erweiterungen zu SysML können in der Arbeit [18] unter „Part II“ nach verfolgt werden.

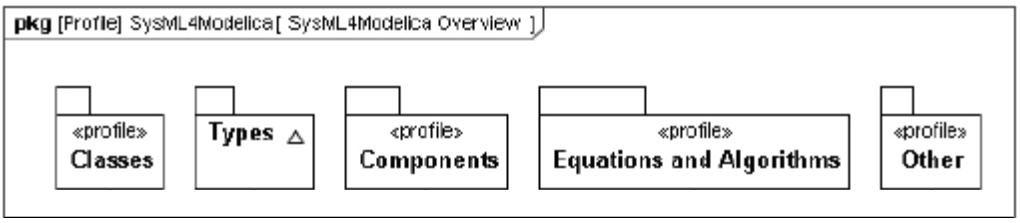


**Abbildung 2-2 Packet-Diagramm als mit einer Übersicht des SysML4Modelica Profils**

Ein nicht standardisiertes Metamodell von Modelica wurde bereits von den Entwicklern selber zur Verfügung gestellt und nennt sich „MetaModelica“ [3].

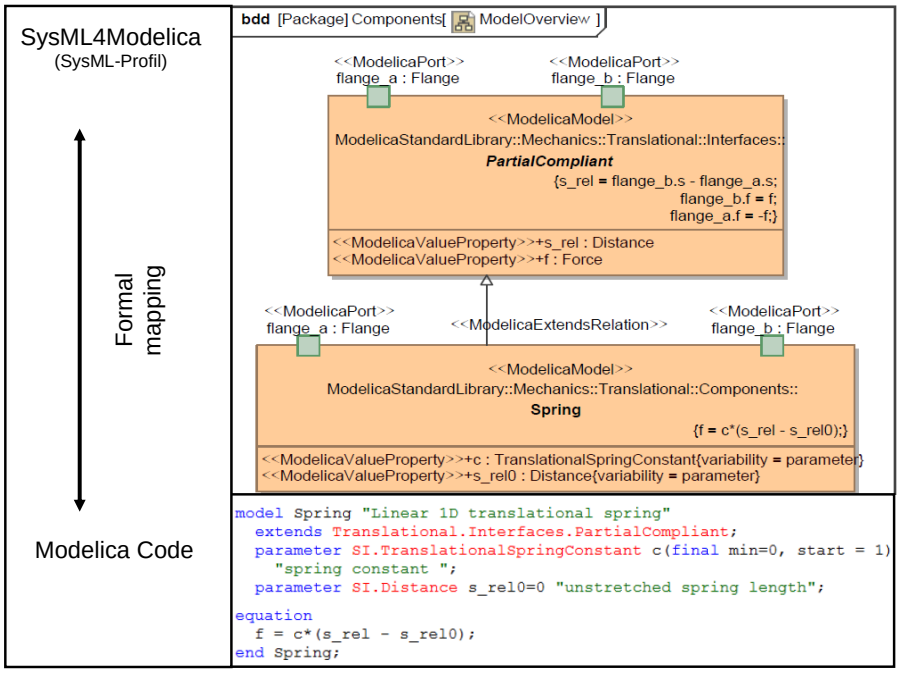


**Abbildung 2-3 Formale Abbildung von Sprachelementen.**

Wie in Abbildung 2-3 zu sehen ist eine formale Abbildung in beide Richtungen gewährleistet.

Die SysML-Modelica Transformation ist eine Referenzimplementierung mit der OMG Query View Transformation (QVT). QVT spezifiziert eine Sprache bzw. Programmiersprache zur Model-zu-Modell Transformation und ist Teil der OMG Meta Object Facilities (MOF) [19]. Somit setzt diese SysML-Modelica Transformation die Konzepte der Model Driven Architektur (MDA) wirksam ein [18].

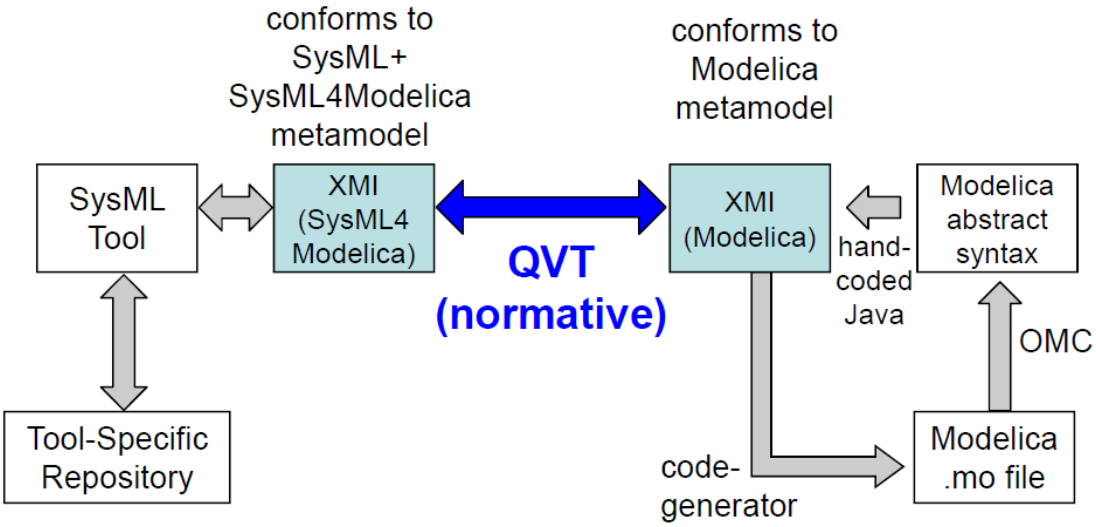


**Abbildung 2-4 Referenzimplementierung der Transformation basierend auf OMG QVT**

### 2.1.2. 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 [20] 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 folgende Struktur- und Verhaltensdiagramme bei der Entwicklung seines Modells zu verwenden:

- Class Diagramm: Repräsentation der Struktur und Beziehungen zwischen Klassen
- Composition Structure Diagram: Repräsentation von Komponenten und deren Beziehung untereinander.
- Activity Diagram: Zur Modellierung von Bedingensgleichungen und algorithmischen Ausdrücken.
- State Machine Diagram: Zur Modellierung von zustandsbasiertem Verhalten.
- Sequenze Diagramme: Zur Modellierung von Interaktionen zwischen einzelnen Systemkomponenten

Ebenfalls gibt es die Möglichkeit Anforderungen textuell zu repräsentieren und diese an Klassen zu binden um deren Einhaltung/Verletzung, beispielsweise während einer Simulation, zu überwachen. Unten links ist die grafische Anforderungsmodellierung als Zustandsautomat und rechts die Verletzung einer Anforderung während einer Simulation durch rote Balken dargestellt.

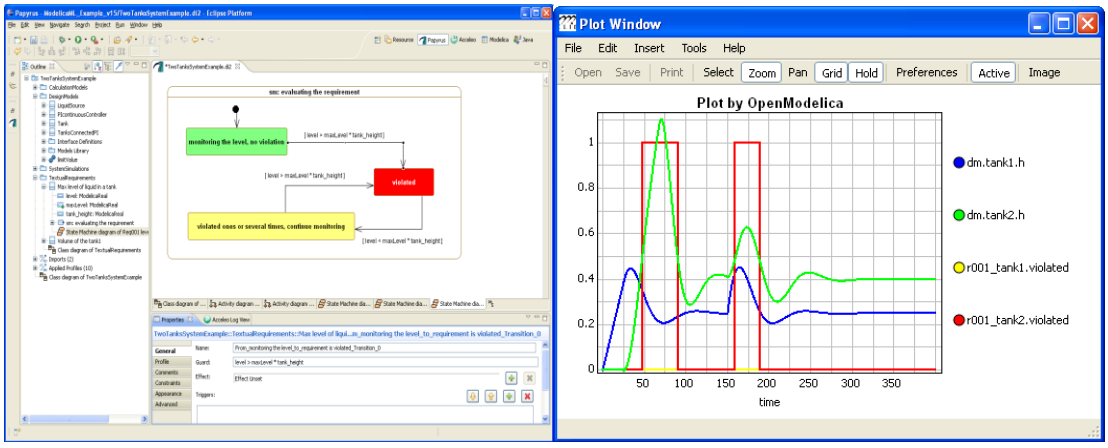


**Abbildung 2-5 Anforderungsmodellierung (links), Repräsentation bei Simulation (rechts)**

Alle entwickelten Stereotypen können auf der Internetpräsenz zu OpenModelica eingesehen werden [20].

## *2.2. Grafische Modellierungsumgebung*

Ein Ziel dieses Projektes ist es offene und damit wiederverwendbare und nicht kommerzielle Lösungen zu Entwickeln. Hierfür bietet sich die Entwicklung von Werkzeugen und Profilen unter Verwendung der Eclipsetechnologie [13] an. Nachfolgend werden daher zwei identifizierte Eclipse-Erweiterungen als sogenanntes „Eclipse Plug-In“ aufgeführt, welche als Modellierungsumgebung, bzw. als Basis dieser, in Frage kommen.

### 2.2.1. TOPCASED

Durch immer komplexere Anforderungen und steigenden Kosten im Bereich der Forschung und Entwicklung, speziell in der Luft- und Raumfahrt, wurde eine Arbeitsgruppe der Airbus France gegründet, welche eine quelloffene und nicht kommerzielle Entwicklungsumgebung für kritische Systeme entwickeln sollte [21]. Das “Toolkit in OPen-source for Critical Applications & SystEms Development”, kurz TOPCASED, ist eine auf der Eclipse-IDE basierende Entwicklungsumgebung zur Unterstützung des MBSE Vorganges [6]. Es ist eine Werkzeugsammlung welche den gesamten Entwicklungsprozess von der Systemspezifikation bis zur Implementierung in Hard- und Software abdeckt. Unter Verwendung der UML zur Modellierung, bietet es auch die Möglichkeit, durch Einbindung von eigenen Zusatztools, die Simulation, Implementierung, Test und Validierung eines Systems zu integrieren. Die aktuelle Version 3.4 bietet bereits viele Erweiterungen welche ebenfalls quelloffen und wieder verwendbar sind. Eine Sammlung der wichtigsten Erweiterungen und Anleitungen zu deren Nutzung findet man unter dem Menüpunkt „Documents“ auf der TOPCASED Website [21].

### 2.2.2. Papyrus

Das Papyrus Eclipse Projekt ist ein nicht kommerzielles, auf der Eclipse Model Development Tools (MDT) basierendes Plug-In [22]. Der Code ist frei verfügbar und nutzt UML 2 als grafische Modellierungssprache. Papyrus bietet dem Benutzer die Möglichkeit das Metamodel seiner gewünschten Modellierungssprache zu implementierten, diesem Metamodell grafischen Elementen zuzuordnen und anschließend eine eigene Modellierungsumgebung zu entwickeln in der die Modellierung mit einem Editor und die Konfiguration der Elemente beispielsweise mit Formularen vorgenommen werden kann.

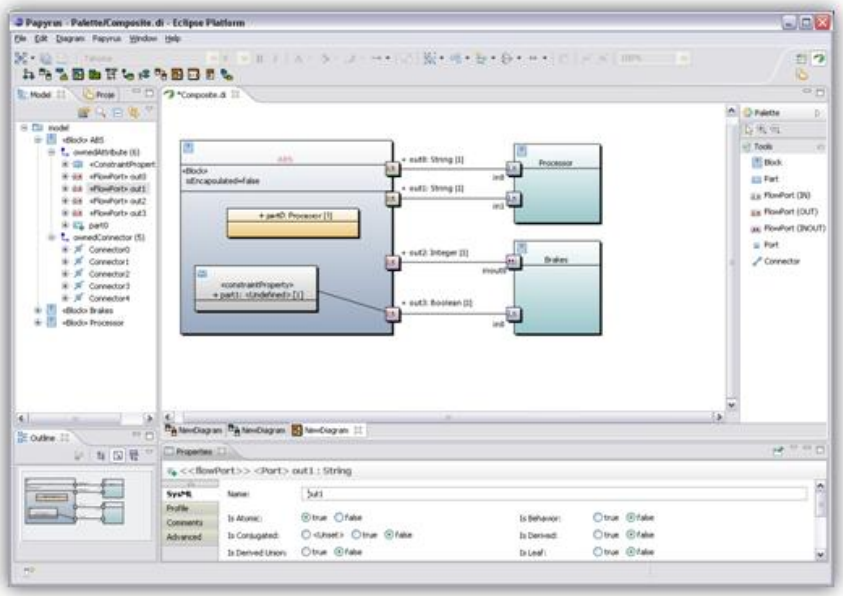

**Abbildung 2-6 Papyrus Modellierungsumgebung**

Grundsätzlich sind folgende Schritte zur Modellierung eines Modells mit einer eigenen Modellierungssprache in Papyrus nötig:

- Das Metamodel der gewünschten Modellierungssprache wird als UML2 Profile für Papyrus entwickelt. Hierzu wird der von Papyrus bereit gestellte „Metamodell Editor“ verwendet.
- Zur Entwicklung der Modellierungsumgebung bietet Papyrus ebenfalls eine Vorlage der UML2 Modellierungsumgebung. Formulare, Kontextmenüs und Assistenten können an die Bedürfnisse eines Modelica Modells angepasst werden.
- Zur Code Generierung wird ebenfalls ein Eclipse Plug-In verwendet. Acceleo ist ein Tool zur Modell-zu-Text-Transformation, welches den Model Driven Architekture (MDA) Ansatz der OMG verfolgt [23]. Unter Verwendung einer Skriptsprache kann das als XML hinterlegte Modell in den gewünschten Code transformiert werden. Dieses Tool lässt sich ohne Probleme in Papyrus integrieren und dort beispielsweise in einem Kontextmenü aufrufen.

## *2.3. System Simulation*

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). Vorher muss der

Modelica Code allerdings noch mit Hilfe eines Compilers in eine Ausführbare bzw. Maschinennahe Sprache übersetzt werden.

Das einzige quelloffene und nicht kommerzielle Tool was für diese Aufgabe in Frage kommt ist OpenModelica (OM) [12]. Daher wurde es schon in der Projektbeschreibung [15] und im ITEA2 als zu verwendendes Werkzeug gefordert [14]. OpenModelica ist eine eigenständige Applikation und kein Eclipse plug-in, somit basiert die Simulationsumgebung auf Eclipse und die Simulationslaufzeitumgebung auf OM. Da zwei unterschiedliche Applikationen nicht ohne weiteres den gleichen Speicher zum Informationsaustausch verwenden können muss ihre Kommunikation separat implementiert werden.

Nachfolgend wird ein Projekt beschrieben, indem eine modifizierte Version von OM als Grundlage einer Laufzeitumgebung verwendet wurde.

### 2.3.1. TCP/IP basiertes Protokoll zur Kommunikation einer Echtzeitsimulation

Das Unternehmen MathCore Engineering AB, bekannt durch ihr MathModelica Tool [24] verwendet ebenfalls Modelica zur Modellierung und Simulation ihrer Modelle. Neben dem kommerziellen Simulationstool MathModelica wird auch eine freie verfügbare Version angeboten, die allerdings auf OpenModelica basiert. Zur Lösung des Kommunikationsproblems zwischen der Simulationsumgebung und der Laufzeitumgebung wurde ein TCP/IP basierter Ansatz entwickelt, um Befehle und Daten austauschen zu können [25].

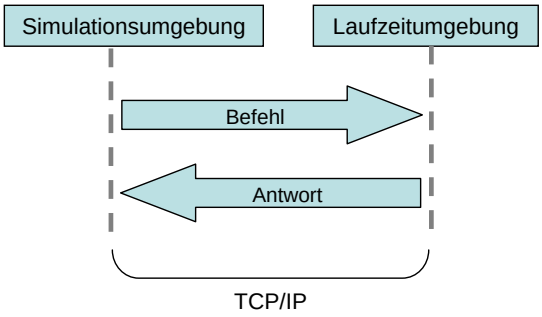


**Abbildung 2-7 Simulationsumgebung (links) und Laufzeitumgebung (rechts)**

Folgende Daten über das zu simulierende Modell werden ausgetauscht:

- Modell Daten (Name, Version, …)
- Variablen (Name, Wert)
- Parameter (Name, Wert)
- Input und Output Variablen (Name, Wert) [Sind Hardware- Signale]

Von MathCore identifizierte Befehle zur Steuerung der Simulationslaufzeitumgebung:

- Start, Stopp, Pause
- Setzen der Werte bei Input Variablen und Parameter
- Setzen der Initialwerte

Die Befehle zur Steuerung der Laufzeitumgebung werden als Zeichenkette in Klartext übermittelt. Als Beispiel das setzen von Parameter Werten:

„`setParameter({"p1", 34.5, "p2", 45.6})`“.

Die zu übermittelnden Simulationsergebnissen werden ebenfalls als Zeichenkette, mit Variablen Name zur Identifikation übermittelt:

„`result("speedSensor.w"=0.1, …, "signalVoltage.v"=20)`“.

Bei der Übermittlung von Simulationsergebnissen ist neben der Menge der zu übermittelnden Daten auch die Frequenz der Übermittlung entscheidend. Zwar erzeugt der Solver kontinuierlich Rechenergebnisse, dennoch bleiben viele der gerechneten Werte gleich oder ändern sich nur bei neuen Eingabewerten.

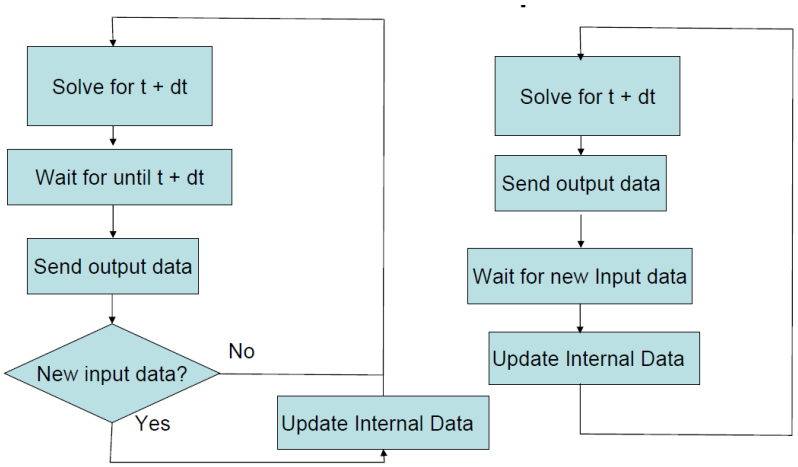


**Abbildung 2-8 Kontinuierliche (links) und eingabeabhängige Übermittlung (recht)**

Abbildung 2-8 zeigt zwei Lösungsansätze. Links werden kontinuierlich Daten übermittelt sobald der Gleichungslöser einen Rechenschritt beendet hat. Rechts werden nur Daten übermittelt, falls der Benutzer eine neue Eingabe getätigt und somit die bestehenden Eingabewerte verändert hat. Welches der beiden Verfahren letztendlich eingesetzt wird, ist allerdings aus der Dokumentation nicht ersichtlich [25].

# 3. Zusammenfassung

Die modelbasierte Systementwicklung mit Unterstützung von UML/SysML und Modelica zeig großes Potenzial die Probleme während der Systementwicklung durch einheitliche und freizugängliche Methoden und Applikationen zu lösen. Da diese Idee allerdings noch sehr jung ist, sind die hier vorgestellten Projekte teilweise selbst noch Forschungsprojekte und könnten sich somit als nicht praktikabel herausstellen.

Die unter 2.1 vorgestellten grafischen Modellierungssprachen zeigen, dass die Modellierung mit UML/SysML und Modelica möglich und bei moderatem Aufwand erlernbar und auch praktikabel einsetzbar ist. Das unter 2.1.1 vorgestellte Projekt verdient dabei besonders große Aufmerksamkeit, da die Entwickler zu der Elite im Bereich der modellbasierten Systementwicklung gehören.

Als Modellierungsumgebung kommen zwar beide unter 2.2 vorgestellten Applikationen in Frage, doch hat sich das TOPCASED, Punkt 2.2.1, in der aktuellen Version 3.4 als kompliziert und an manchen Stellen schlecht dokumentiert erwiesen, was den Einsatz und die Weiterentwicklung erschwert. Papyrus, Punkt 2.2.2, wurde bereits in Verbindung mit ModelicaML getestet und bot eine gute und solide Plattform für weitere Arbeiten.

Das vorgeschlagene Kommunikationsprotokoll der MathCore Entwickler wie unter 2.3 beschrieben, zeigt Lücken und wurde auch auf Nachfrage nicht weiter erläutert. Es ist dennoch für den Einsatz im nichtinteraktiven sowieso interaktiven Simulationsbereich zu berücksichtigen. Ein ähnlicher und an vielen Stellen ausgereifterer Vorschlag ist in der Arbeit [16] zu finden, wo auch eine vollständige Spezifikation zu einem Protokoll zu finden ist.

Für den weiteren Projektverlauf werden verwandte Projekte im Bereich der Simulationsumgebung und dem Testen identifiziert und die hier vorgestellten Projekte in der Praxis erprobt.

# 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: 2009 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, (Internet 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] 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.

[18] 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

[19] OMG Meta Object Facility (MOF) 2.0 Query/View/Transformation, v1.0, Last Accessed: 2010, http://www.omg.org/spec/QVT/1.0/

[20] 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

[21] TOPCASED Homepage, Last Accessed: 2010 http://www.topcased.org

[22] Papyrus for UML, Last Accessed: 2009 http://www.papyrusuml.org/

[23] Acceleo Eclipse Plug-In, Last Accessed: 2010 www.acceleo.org/pages/home/en

[24] MathCore Engineering AB, MathModelica 2.1, Last Accessed: 2010 http://www.mathcore.com/

[25] Otto Tronarp, “A TCP/IP Based Protocol for Communicating with Real Time Simulations, modprod2010-Day1-OttoTronarp-TCP-IP-protocol-for-simulation.pdf, 2009