T U M

I N S T I T U T F Ü R I N F O R M A T I K

Stand und Anforderungen an eine Werkzeugunterstuetzung zur Entwicklung von Automatisierungssoftware

Dominik Sojer und Christian Buckl und Alois Knoll

TUM-I1003
Februar 10

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-02-I1003-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Kurzfassung

Der Bericht entstand im Rahmen des, vom BMBF geförderten Projekts „Software Platform Embedded Systems 2020 (SPES2020)“ im Teilprojekt Automatisierungstechnik. Wir bedanken uns bei den Industriepartnern Embedded4You und Siemens für Ihre Unterstützung bei der Erstellung dieses Dokuments.

Dieses Dokument beschreibt die spezifischen Anforderungen an Entwicklungs- und Testwerkzeuge zur Modellgetriebenen Softwareentwicklung der Partner des Anwendungsgebiets Automatisierungstechnik, welche durch die aktuell verfügbaren Werkzeuge noch nicht erfüllt werden. Dabei werden projektorganisatorische Defizite behandelt, aber hauptsächlich werden konkrete Funktionen zum Umgang mit Modellen diskutiert, die zurzeit noch nicht zum Stand der Technik in industriell einsetzbaren Werkzeugen gehören.

Darüber hinaus werden Beispiele präsentiert, welche Entwicklungs- und Testwerkzeuge im Moment von den beteiligten Industriepartnern im Anwendungsgebiet Automatisierungstechnik eingesetzt werden.

Inhalt

1 Einordnung und Kurzbeschreibung

1.1 Motivation und Einordnung

Der Bericht entstand im Rahmen des, vom BMBF geförderten Projekts „Software Platform Embedded Systems 2020 (SPES2020)“ im Teilprojekt Automatisierungstechnik. Wir bedanken uns bei den Industriepartnern Embedded4You und Siemens für Ihre Unterstützung bei der Erstellung dieses Dokuments.

Das vorliegende Dokument soll aufzeigen, welche Anforderungen an modellgetriebene Entwicklungs- und Testwerkzeuge in der Automatisierungstechnik bestehen, die von den aktuell am Markt verfügbaren Werkzeugen nicht erfüllt werden.
Im SPES Gesamtkontext sollen somit dem Zentralprojekt die Bedürfnisse dieser Ingenieursdisziplin, hinsichtlich modellgetriebener Entwicklung, näher gebracht werden, damit diese Bedürfnisse in den verschiedenen Arbeitsgruppen des Zentralprojekts berücksichtigt werden können.

1.2 Management Summary

Dieses Dokument beschreibt die spezifischen Anforderungen an Entwicklungs- und Testwerkzeuge zur Modellgetriebenen Softwareentwicklung der Partner des Anwendungsgebiets Automatisierungstechnik, welche durch die aktuell verfügbaren Werkzeuge noch nicht erfüllt werden. Dabei werden projektorganisatorische Defizite behandelt, wie etwa die bisher fehlende Berücksichtigung bereits bestehender Entwicklungsprozesse oder die bisher nur mangelhaft umgesetzte Versionsverwaltung von Modellen, aber hauptsächlich werden konkrete Funktionen zum Umgang mit Modellen diskutiert, die zurzeit noch nicht zum Stand der Technik gehören. Viele dieser Kritikpunkte befassen sich mit dem Problem der konsistenten Bearbeitung von Modellen über Werkzeuggrenzen und Abstraktionsebenen hinweg. So ist für die Kopplung der vorhandenen Werkzeuge eine klare Definition der Schnittstellen der Werkzeuge erforderlich, um durchgängig Informationen zwischen Modellen austauschen zu können.

Des Weiteren wird von den Partnern aufgezeigt, dass aktuelle Modellgetriebene Entwicklungswerkzeuge nicht mächtig genug sind, um automatisierungstechnische Systeme umfassend zu beschreiben. Beispielsweise fehlt häufig eine passende Möglichkeit zur Modellierung von Zeit und zur hybriden Modellierung des Systemverhaltens.

Neben der Aufzählung von Defiziten werden, sowohl für Entwicklungs-, als auch für Testwerkzeuge, Beispiele genannt, die bei den Industriepartnern im Einsatz sind und die, in diesem Dokument beschriebenen Anforderungen nicht erfüllen.

1.3 Überblick

Kapitel 2 beschreibt die besonderen Eigenschaften der Automatisierungstechnik, die sie und ihre Entwicklungsprozesse von anderen Ingenieurswissenschaften abgrenzen. Kapitel 3 listet Anforderungen an Entwicklungswerkzeuge auf, die von den Industriepartnern im Teilprojekt „Automatisierungstechnik“ gestellt werden, die allerdings von den aktuell am Markt verfügbaren Werkzeugen nicht erfüllt werden. Kapitel 4 bietet einen nicht repräsentativen Überblick über Entwicklungswerkzeuge, die bei den Industriepartnern aktuell zur Entwicklung von automatisierungstechnischen Systemen eingesetzt werden. Kapitel 5 beschreibt, analog zu Kapitel 3, besondere Anforderungen der Automatisierungstechnik an Testwerkzeuge. In Kapitel 6 werden

abschließend Testwerkzeuge aufgelistet, die aktuell bei den Industriepartnern im Einsatz sind.

2 Besonderheiten der Automatisierungstechnik

Bei der Automatisierungstechnik handelt es sich um eine Ingenieurwissenschaft, die das Ziel verfolgt, Anlagen möglichst automatisch, d.h. ohne Steuerung und Interaktion durch Menschen, zu betreiben. Die wichtigsten Ziele, die mit Hilfe der Automatisierungstechnik erreicht werden sollen, sind zum einen die Senkung von Produktionskosten durch die Einsparung von Personalkosten, und zum anderen die Steigerung der Produktqualität durch die Vermeidung menschlicher Fehler im Produktionsprozess. Das Fachgebiet Automatisierungstechnik zeichnet sich durch eine sehr große Heterogenität der Zielsysteme aus: einerseits werden kleine, Speicherprogrammierbare Steuerungen eingesetzt, andererseits stellen aber auch ganze Anlagen, wie Fließbänder und Produktionsstraßen typische Systeme der Automatisierungstechnik dar. Aufgrund dieser starken Heterogenität und der dadurch verbundenen Querschnittsfunktion durch verschiedene andere Wissenschaften stellt die Automatisierungstechnik eine Reihe von besonderen Anforderungen an Werkzeuge zur Modellgetriebenen Softwareentwicklung.

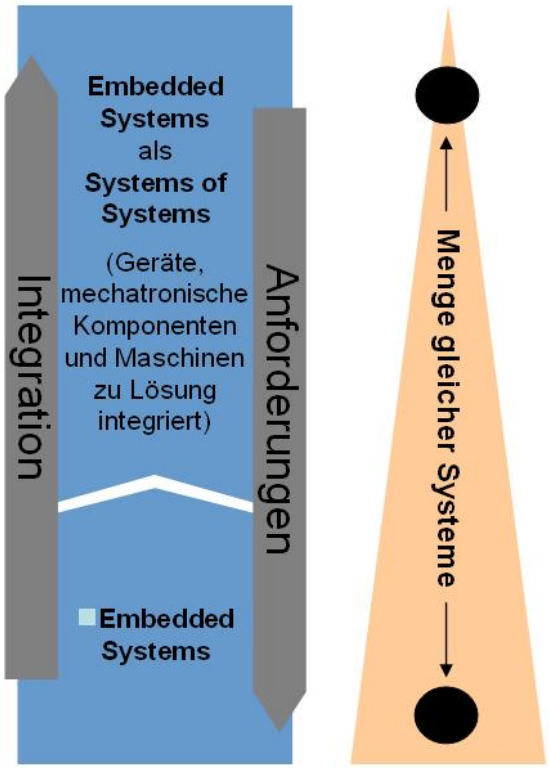
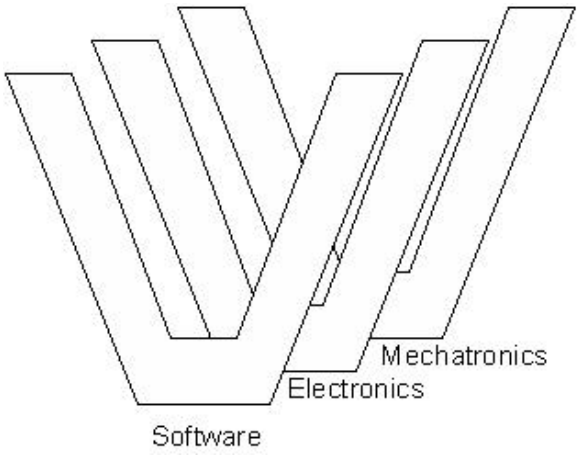


Bild 1: Verschiedene Abstraktionsebenen in der Automatisierungstechnik

Die großen Anlagen, mit denen sich die Automatisierungstechnik beschäftigt können als Spitze einer Abstraktionshierarchie betrachtet werden. Sie sind oft Unikate oder werden nur in sehr geringen Stückzahlen gefertigt. Sie können allerdings aus Standardkomponenten bestehen, die wiederum das untere Ende der Abstraktionshierarchie darstellen. Zwischen diesen beiden Extremen gibt es noch eine Reihe weiterer

solcher Ebenen. Beispiele hierfür wären Produktionszellen oder Roboter inkl. Steuerungsgeräten.
Hieraus ergeben sich zwei zentrale Sichtweisen auf das Gesamtsystem. Zum einen die Sichtweise der mehrstufigen Integration von kleinen Bauteilen zu immer größeren Komponenten, zum anderen die Sichtweise der Anforderungen, die vom Gesamtsystem in die Subsysteme getragen werden. Dies wird in Bild 1 veranschaulicht.
Eine weitere wichtige Besonderheit der Automatisierungstechnik ist die Verknüpfung verschiedenster Domänen. Während in der Entwicklung von Eingebetteten Systemen schon Hardware und Software gemeinsam betrachtet werden müssen, so kommen in der Automatisierungstechnik noch eine Reihe weiterer Disziplinen, wie zum Beispiel Mechanik und Verfahrenstechnik, hinzu.
Alle diese Einzelwissenschaften besitzen ihre eigenen Denk- und Vorgehensweisen, weshalb eine Vereinigung der Entwicklungsprozesse, die in diesen Disziplinen vorherrschen, als praktisch nicht durchführbar gilt. Stattdessen ist es sinnvoll, den Entwicklungsprozess jeder Disziplin als eigene Dimension eines mehrdimensionalen V-Modells zu betrachten, zwischen denen Abhängigkeiten bestehen und die sich zu verschiedenen Zeitpunkten während des Entwicklungsprozesses synchronisieren.



Diese Abstimmung zwischen den verschiedenen Einzelwissenschaften stellt eine große Herausforderung dar, da an den Schnittstellen die Gefahr besteht, Informationen zu verlieren oder sie im disziplinspezifischen Kontext falsch zu interpretieren. Eine Standardisierung oder gar Automatisierung dieser Informationsaustausche wäre deswegen sehr wünschenswert, aber aufgrund der hohen Komplexität davon existieren bisher keine entsprechenden Lösungen.

3 Anforderungen aus Sicht des Anwendungsgebiets Automatisierungstechnik an Entwicklungswerkzeuge

Die modellgetriebene Softwareentwicklung ist ein Ansatz zur Softwareentwicklung, bei dem die Arbeit der Entwickler nicht im Schreiben von Quellcode besteht, sondern in der Modellierung von Anwendungen. Erst in einem zweiten Schritt wird aus diesen Modellen automatisch Quellcode generiert. Diese Erhöhung des Abstraktionsgrads in der Softwareentwicklung soll zum einen den Entwicklungsprozess beschleunigen und zum anderen die Softwarequalität erhöhen, indem vermieden wird, dass Entwickler Arbeitsschritte wiederholen müssen. Dadurch soll sich die Anzahl von Programmierfehlern verringern.

Die Idee, Quellcode automatisch generieren zu lassen, ist nicht neu, sondern sie wurde schon vor 20 Jahren von den so genannten CASE-Werkzeugen (=“Computer Aided Software Engineering“) aufgegriffen. Allerdings hat sich der Versuch, Software möglichst vollständig automatisch generieren zu lassen, als sehr unflexibel herausgestellt, weshalb moderne Ansätze, die unter dem Begriff Modellgetriebene Softwareentwicklung („MDSD“) firmieren, nur versuchen, häufig wiederkehrende Programmkomponenten automatisch zu generieren, was je nach Anwendung ein unterschiedlicher großer Teil der Software sein kann [1]. Um dabei möglichst flexibel auf verschiedenste Problemstellungen in einem bestimmten Anwendungsbereich eingehen zu können, kann es oft notwendig sein, vor der expliziten Modellierung des Programms eine entsprechende domänenspezifische Sprache für diesen Bereich als Metamodell zu definieren.

Offensichtlich ist modellgetriebene Softwareentwicklung allerdings nur effizient einsetzbar, wenn die Entwickler durch entsprechende Werkzeuge unterstützt werden, welche die Erstellung eines passenden Metamodells für die Anwendung, die Modellierung von Software und die Generierung von Quellcode erlauben. Zwar existiert mittlerweile eine breite Palette solcher Werkzeuge, allerdings schaffen diese es nicht, sämtliche an sie gerichteten Anforderungen zu erfüllen.

Anforderungen, welche bisher von keinen der existierenden Werkzeuge erfüllt werden, werden im Folgenden dargestellt. Diese Anforderungen wurden im Rahmen des Projekts „Software Plattform Embedded Systems 2020“ im Teilprojekt „Anwendungsgebiet: Automatisierungstechnik“ gesammelt, und erheben somit nicht den Anspruch auf Vollständigkeit. Sie spiegeln allein die Defizite des Ist-Zustands der aktuellen Werkzeuge im Bezug auf Anwendungen aus dem Bereich der Automatisierungstechnik dar.

Name	1. Anpassbarkeit an den Entwicklungsprozesses
Beschreibung	Die Modellgetriebene Softwareentwicklung darf keine spezifischen Softwareentwicklungsprozesse voraussetzen, sondern muss flexibel genug sein um in möglichst vielen verschiedenen Entwicklungsprozessen eingesetzt werden zu können.
Motivation	Eine große Schwäche aktueller Werkzeuge zur modellgetriebenen Softwareentwicklung ist, dass sie nicht auf das organisatorische Umfeld aktueller Softwareentwicklung eingehen. So wird von vielen dieser Werkzeuge ein sehr spezifischer Softwareentwicklungsprozess vorausgesetzt, um effizient modellgetrieben Software zu entwickeln.

	Das stellt sich in der Realität aber als sehr problematisch heraus, da die existierenden und realisierten Softwareentwicklungsprozesse damit häufig nicht vereinbar sind. Beispielsweise setzen sich Stahl und Völter in [1] stark für eine ausschließliche Verwendung Modell-getriebener Entwicklungsmethoden in spiralförmigen Entwicklungsprozessen ein in denen bei jeder Iteration auch die Werkzeuge selbst verändert werden sollen. Allerdings folgen viele Softwareentwicklungsprojekte dem V-Modell und Wasserfallmethoden, die so ein Vorgehen nicht unterstützen. Die Umstellung eines bewährten Softwareentwicklungsprozesses kann allerdings mit hohen Kosten und Risiken verbunden sein und ist deshalb meist nicht erstrebenswert, weshalb die einfachere Lösung darin bestehen würde, dass die Werkzeuge zur modellgetriebenen Softwareentwicklung sich besser in klassische Softwareentwicklungsprozesse einfügen lassen.

Name	2. Versionsmanagement
Beschreibung	Die Versionsverwaltung von Modellen darf sich nicht auf eine klassische, textuelle Versionverwaltung beschränken sondern muss auf Syntax und Semantik der Modelle Bezug nehmen.
Motivation	Ein Punkt in dem die Differenz zwischen einem modellgetriebenen Softwareentwicklungsprozess und herkömmlichen Softwareentwicklungsprozessen sehr deutlich wird, ist die Versionierung der grundlegenden Programmbausteine, also einerseits Quellcode und andererseits Software Modelle. Während in der klassischen Softwareentwicklung die Versionsverwaltung von Quellcode ein grundlegendes Mittel zur Sicherung und Überwachung des Entwicklungsfortschrittes und zur Ermöglichung eines, eventuell über Standortgrenzen hinweg, verteilten Entwicklungsprozesses ist, greifen aktuelle Werkzeuge zur modellgetriebenen Softwareentwicklung dieses Konzept noch nicht auf. Somit eignen sie sich zurzeit nicht sehr gut zur Softwareentwicklung durch verteilte Entwicklerteams. Problematisch ist, dass Änderungen an Modellen häufig Auswirkungen auf mehrere Dateien mit sich bringen können, die noch dazu oft nicht mit einfachen DIFF/MERGE- Strategien auf textueller Basis wieder zusammengeführt werden können. Das Sperren aller Modell-Dateien, auf die ein Entwickler gerade zugreift, ist ebenfalls keine Lösung, da somit der Entwicklungsprozess sequentialisiert wird. Aus Sicht der Softwareentwickler ist es essentiell, das Werkzeuge zur modellgetriebenen Softwareentwicklung dieses Problem lösen, um diesen Ansatz auch in größeren Entwicklungsprojekten einzusetzen.

Name	3. Wiederverwendung
Beschreibung	Die projektübergreifende Wiederverwendung von Artefakten der mo-

	dellgetriebenen Softwareentwicklung muss ermöglicht werden. Dabei muss beachtet werden, dass in der Automatisierungstechnik auf verschiedenen Abstraktionsebenen entwickelt wird, auf denen die Wiederverwendung jeweils anderen Rahmenbedingungen unterliegt (z.B. Hardwareabhängigkeit auf sehr tiefen Ebenen).
Motivation	Ein weiteres Ziel aktueller Softwareentwicklung ist die projektübergreifende Wiederverwendung von Software-Komponenten, welche in der klassischen Softwareentwicklung durch die große Verbreitung von objektorientierten Programmiersprachen sehr populär wurde, da hierbei dieses Ziel sehr leicht erreicht werden kann. Die Verwendung von hierarchischen, modularen Komponenten, die projektübergreifend eingesetzt werden können, stellt ein sehr effektives Mittel zur Erhöhung der Softwarequalität bei sinkender Entwicklungszeit dar. Wegen diesen positiven Eigenschaften wäre ein ähnliches Konzept auch für die Modellgetriebene Softwareentwicklung wünschenswert, allerdings hat noch kein aktuelles Werkzeug eine zufrieden stellende Umsetzung dieser Anforderung erreicht.

Neben diesen projektorganisatorischen Mängeln der verfügbaren Werkzeuge zur modellgetriebenen Softwareentwicklung besteht auch eine Reihe von fachlichen Anforderungen an sie, denen sie zum aktuellen Zeitpunkt noch nicht gerecht werden.

Name	4. Hybride Modelle
Beschreibung	Modelle sollten sowohl kontinuierlich als auch diskrete Wertebereiche abbilden können.
Motivation	Ein sehr deutlicher, fachlicher Mangel ist die fehlende Unterstützung von hybriden Modellen, die sowohl diskrete, als auch kontinuierliche Werte beinhalten. Die Notwendigkeit für solch eine Unterstützung mag auf den ersten Blick nicht ganz klar sein, allerdings handelt es sich bei sehr vielen eingebetteten Systemen, die in der Automatisierungstechnik zum Einsatz kommen, in der Realität um hybride, prozesstechnische Systeme, die mit Hilfe von diskreten Berechnungen über Sensoren und Aktoren auf ihre kontinuierliche Umwelt reagieren müssen. Die Unterstützung dieser Hybridität auf Modellebene würde die Entwicklung dieser eingebetteten Systeme stark vereinfachen. [3]

Name	5. Zeitmodellierung
Beschreibung	Viele zeitliche Aspekte, wie z.B. Jitter, können mit einem klassischen Zeitmodell nicht beschrieben werden.

Motivation	Ein weiterer Punkt, an dem aktuelle Werkzeuge zur modellgetrieben Softwareentwicklung sehr schnell an ihre Grenzen stoßen, ist die Modellierung der Zeit, welche in allen reaktiven Systemen und insbesondere in sicherheitskritischen Systemen eine wichtige Rolle spielt. Zwar erlauben Modellierungssprachen, wie die UML, die Modellierung von zeitlichen Abläufen, allerdings genügen die Umsetzungen dieser Konzepte bisher nicht den an sie gestellten Anforderungen, wie sie zum Beispiel von zeitgesteuerten Bussystemen vorgegeben werden. Dabei reicht es häufig nicht aus, die bloße Abfolge von Systemzuständen in endlichen Automaten zu modellieren, sondern es müssten detaillierte Informationen über die Zeit in die Modellierung einfließen (z.B. Jitter), wie es zum Beispiel theoretisch mit Timed Automata möglich wäre. [2] Eine bessere Unterstützung der Modellierung von Zeit durch die Entwicklungswerkzeuge zur modellgetriebenen Softwareentwicklung würde somit die Generierung von weiteren nicht-funktionalen Programmkomponenten ermöglichen und das Ziel, dem Entwickler gleichartige, sich wiederholende Tätigkeiten zu ersparen, unterstützen.

Name	6. Kombination von Struktur und Verhalten
Beschreibung	Modellelemente sollten sowohl Struktur als auch Verhaltensbeschreibungen beinhalten. Diese Anforderung ist sehr stark mit Anforderung 9 „Integration von Sichten“ verknüpft, beschreibt aber einen anderen Blickwinkel. Hier steht klar die einzelne Komponente im Fokus und nicht das Gesamtsystem.
Motivation	Eine sinnvolle Unterstützung der Modellierung zeitlicher Abläufe stellt eine grundlegende Eigenschaft dar, um einzelne Modellkomponenten zu „technologischen Komponenten“ werden zu lassen, welche sowohl eine Struktur- als auch eine Verhaltensbeschreibung beinhalten. Eine technologische Komponente ist ein Teilsystem und eine wieder verwendbare Komponente (z.B. ein Motor), das potentiell durch verschiedene Sichten (z.B. Verhalten, Mechanik, Elektronik) beschrieben werden kann. Das Ziel der Wiederverwendbarkeit von Modellkomponenten würde sehr stark unterstützt werden, wenn Werkzeuge zur modellgetriebenen Softwareentwicklung dieses Konzept umsetzen würden und somit Struktur und Verhalten auf gleicher Ebene betrachtet werden könnten. Darüber hinaus könnten bei der Kombination verschiedener „technologischer Komponenten“ automatische Tests durchgeführt werden, die prüfen ob deren Verhalten sich negativ beeinflusst. Somit könnten frühzeitig im Entwicklungsprozess Probleme und Fehler aufgedeckt werden, deren Behebung dabei weitaus günstiger ist, als eine Fehlerbehebung in einem späteren Entwicklungszustand.

Name	7. Offene Schnittstellen
Beschreibung	Werkzeuge zur modellgetriebenen Softwareentwicklung sollten offene Schnittstellen bereitstellen, um die Ergebnisse eines Entwicklungsschrittes verlustfrei in den nächsten Schritt übertragen zu können.
Motivation	Neben dem Fehlen von zeitlichen Aspekten in der Modellierung von eingebetteten Systemen ist ein weiteres Manko aktueller Werkzeuge zur modellgetriebenen Softwareentwicklung das Fehlen offener Schnittstellen. Nachdem kein aktuell verfügbares Werkzeug für sich in Anspruch nehmen kann, alle Bereiche des Softwareentwicklungsprozesses abzudecken, sind offene Schnittstellen ein möglicher Weg um vorhandene Informationen über die verschiedenen Phasen eines Entwicklungsprozesses zu transportieren, und somit die Konsistenz und Vollständigkeit der verschiedenen Phasen- und Werkzeugübergänge zu gewährleisten. Ein Spezialfall davon stellt das so genannte „Tracing“, also die Verfolgung von Systemanforderungen dar. In einem klassischen Software Entwicklungsprozess werden in einem ersten Schritt solche Anforderungen identifiziert und fließen dann in den Entwicklungs- und Testprozess ein. Häufig ändern sich allerdings solche Anforderungen während des Entwicklungsprozesses, weshalb eine möglichst einfache Konsistenz- und Vollständigkeitsprüfung zu jeder Zeit sehr wünschenswert ist. Werkzeuge zur modellgetriebenen Softwareentwicklung müssen also die Möglichkeit bieten, mit solchen Anforderungen umzugehen. Neben der Bereitstellung von Schnittstellen zum syntaktischen Austausch von Daten, ist die Unterstützung einer semantischen Integration wünschenswert.

Name	8. Laufzeitmodelle
Beschreibung	Ein System sollte zur Laufzeit Kenntnis über seine eigene Struktur besitzen, um jederzeit eine aktuelle Systemdokumentation zur Verfügung stellen zu können und Rekonfigurationen zu ermöglichen. Dabei muss aber der Schutz von geistigem Eigentum gewahrt bleiben.
Motivation	In manchen Situationen kann es sehr sinnvoll sein, die Möglichkeit zu besitzen**,** Informationen über die Struktur eines Systems nicht nur durch die verschiedenen Phasen des Entwicklungsprozesses zur Verfügung zu stellen, sondern sogar darüber hinaus auch zur Laufzeit. Die Möglichkeit, auf diese Informationen zur Laufzeit in einer geordneten, hierarchischen Form zuzugreifen erlaubt zum einen konsistente Rekonfigurationen des Systems zur Laufzeit und zum anderen wird auch das Problem gelöst, dass die Dokumentation von Systemen, die schon lange im Einsatz sind, sich manchmal nicht mit

	der implementierten Struktur des Systems deckt. Werkzeuge zur modellgetriebenen Softwareentwicklung sollten diese Möglichkeit ebenfalls unterstützen.

Name	9. Integration von Sichten
Beschreibung	Es müssen verschiedene Sichten auf ein Modell ermöglich werden. Außerdem dürfen Änderungen am Modell in einer Sicht nicht zu inkonsistenten Zuständen des Modells in anderen Sichten führen.
Motivation	Eine weitere wichtige Anforderung, die stark mit der Einführung offener Werkzeugschnittstellen verbunden ist, stellt die Möglichkeit dar, verschiedenste Modelle ineinander zu integrieren. Diese Anforderungen leiten sich in der Realität daraus ab, dass im Bereich der Automatisierungstechnik verschiedenste Gewerke zusammenarbeiten müssen, wie etwa Maschinenbau, Elektrotechnik und Informatik. Jede dieser Domänen besitzt ihre eigenen Werkzeuge und Denkweisen, wobei es sich dabei in vielen Fällen nur um verschiedene Sichten auf elementare, zugrunde liegende Komponenten handelt. Beispielsweise haben Struktur- und Verhaltensmodelle eines Systems visuell kaum oder keine Gemeinsamkeiten, wobei sie ein und dieselbe Komponente beschreiben. Ähnlich verhält es sich auch mit mechanischen, elektrotechnischen und informatischen Beschreibungen von Systemen der Automatisierungstechnik. Die Integration dieser verschiedenen Sichten und die konsistente Handhabung der Auswirkung von Änderungen in einer Sicht auf die anderen Sichten, stellt eine wichtige Anforderung an Werkzeuge zur modellgetriebenen Softwareentwicklung dar, da dies momentan noch häufig manuell geschieht und somit eine große Quelle von Fehlern und Inkonsistenzen darstellt.

Name	10. Flexibilität
Beschreibung	Die Werkzeuge zur modellgetriebenen Softwareentwicklung müssen sowohl die Entwicklung von „systems“ als auch die Entwicklung von „systems of systems“ unterstützen.
Motivation	Neben der Heterogenität der, an einem Projekt der Automatisierungstechnik beteiligten Fachdisziplinen, ist eine weitere Schwierigkeit die allgemeine Heterogenität der betrachteten Systeme in Bereich der Automatisierungstechnik. Die dabei verwendeten Entwicklungswerkzeuge sollten sowohl zur Entwicklung von Systemen, als auch zur Entwicklung von „systems of systems“ geeignet sein. Dabei müssen sie mit erheblichen Unterschieden der betrachteten Komponenten umgehen können, wie zum Beispiel mit sehr großen Unterschieden in der Rechenleistung und dem verfügbaren Speicherplatz. Die Werkzeuge zur modellgetriebenen Softwareentwicklung sollten dem Rechnung tragen und dem Entwickler möglichst wenige Sys-

	temeigenschaften vorgeben, um durch eine hohe Flexibilität auf diese Unterschiede reagieren zu können.

Name	11. Hardware-/Software-Codesign
Beschreibung	Automatisierungstechnische Systeme bestehen nicht nur aus Software, sondern auch aus Hardware und Mechanik. Diese Systemkomponenten sollten auch modelliert werden können.
Motivation	Die Heterogenität der Zielsysteme in der Automatisierungstechnik erstreckt sich darüber hinaus auch auf einzelne Systemkomponenten, die nur bei sehr wenigen Systemen zum Einsatz kommen. So ist es zum Beispiel notwendig, auch sehr hardwarenahe Komponenten modellgetrieben entwickeln zu können, wie etwa Treiber und Busprotokolle, obwohl diese von System zu System sehr unterschiedlich sein können. Diese Anforderung können aktuelle Entwicklungswerkzeuge ebenfalls nicht erfüllen.

Diese Liste mit Anforderungen an Entwicklungswerkzeuge für die modellgetriebene Softwareentwicklung erhebt, wie in der Einleitung bereits geschildert, keinen Anspruch auf Vollständigkeit, da sie in Zusammenarbeit mit den Partnern des Teilprojekts Automatiserungstechnik erhoben wurden und deren Anforderungen widerspiegeln. Je nach Anwendungsfall können noch weitere Anforderungen hinzukommen, oder es können auch manche Anforderungen dieser Liste obsolet werden.

4 Anforderungen aus Sicht des Anwendungsgebiets Automatisierungstechnik an Testwerkzeuge

Neben den besonderen Anforderungen der Automatisierungstechnik an Werkzeuge zur modellgetriebenen Softwareentwicklung gibt es auch noch eine Reihe von speziellen Anforderungen, die diese Domäne an Testwerkzeuge stellt.

Name	1. Kontinuierliche Qualitätssicherung
Beschreibung	Die Testwerkzeuge sollten eine prozessbegleitende Qualitätssicherung ermöglichen um möglichst einfach und früh die Qualität des Systems bewerten zu können.
Motivation	Bei den Arbeitsergebnissen der Automatisierungstechnik kann es sich um komplette industrielle Anlagen mit einer Ausdehnung von mehreren hundert Metern handeln, weshalb es nur schwer möglich ist, dem Kunden vor Fertigstellung bestimmte Systemfunktionen zu verdeutlichen. Wünschenswert wäre deshalb eine prozessbegleitende Qualitätssicherung, die einerseits zur Kommunikation mit dem Kunden verwendet werden kann. Andererseits sollte es damit aber auch möglich sein, bereits während der Systementwicklung bestimmte Funktionen zu validieren und eine virtuelle Inbetriebnahme durchzuführen. Diese Techniken würden es ermöglichen, Fehler im Systementwurf und Missverständnisse in der Kundenkommunikation früher zu erkennen und somit die Kosten für deren Behebung erheb-

	lich zu senken.

Name	2. Unterstützung der Zertifizierung
Beschreibung	Testwerkzeuge sollten Ergebnisse liefern, die möglichst einfach für einen Zertifizierungsprozess verwendet werden können
Motivation	Eine weitere wünschenswerte Eigenschaft für Testwerkzeuge wäre die Möglichkeit, Fehlermodelle zum Testen des Systems automatisch aus Zertifizierungsrichtlinien abzuleiten. Auf die System-Zertifizierung kann in manchen Branchen ein beträchtlicher Anteil der Entwicklungskosten eines Systems entfallen. Bisher gibt es für den Zertifizierungsprozess aber kaum Werkzeugunterstützung, weshalb an dieser Stelle auch ein erhebliches Kosteneinsparpotential besteht, falls die automatische Extraktion von Fehlermodellen aus den Zertifizierungsrichtlinien von den Zertifizierungsbehörden akzeptiert wird. Bisher existieren zwar Werkzeuge, die zertifizierungsrelevanten Output liefern, allerdings muss hierfür das komplette System meist neu modelliert werden (z.B. für Fehlerbaumanalysen).

Name	3. Qualitätsmessung
Beschreibung	Existierende Metriken für Modelle haben häufig eine sehr geringe Aussagekraft, weshalb neue aussagekräftigere Metriken benötigt werden.
Motivation	Darüber hinaus fehlen in den aktuell verfügbaren Werkzeugen auch Mechanismen zur Abschätzung der Qualität von Modellen. In der klassischen Softwareentwicklung gibt es eine Reihe von Metriken, die mit unterschiedlichen Ansätzen versuchen, die Qualität von Software zu bestimmen. Zwar gibt es auch dort noch keinen allgemeingültigen Ansatz, der kritiklos von der Softwarequalitäts-Gemeinde aufgenommen wurde, aber für eine Vielzahl von Software Entwicklungsprojekten ist eine Anwendung von zum Beispiel funktionsorientierten Metriken oder Lines-of-Code Analysen sehr sinnvoll.

5 Zusammenfassung

Mit diesem Dokument sollten die grundlegenden Herausforderungen für die Entwicklungs- und Testwerkzeuge der nächsten Generation zur Modellgetriebenen Entwicklung im Bereich der Automatisierungstechnik dargestellt werden. Diese wurden im Rahmen des Teilprojekts „Automatisierungstechnik“ des Projekts SPES 2020 mit Hilfe der beteiligten Industriepartner identifiziert. Dabei wurden zwei sehr unterschiedliche Bereiche identifiziert, in denen die aktuell verfügbaren Werkzeuge noch große Defizite aufweisen. Zum einen im Bereich der Integration der Werkzeuge in bestehende Entwicklungsprozesse (z.B. die noch nicht zufrieden stellend gelöste Möglichkeit zur Versionierung von Modellen) und zum anderen im Bereich der Mächtigkeit der verwendeten Modelle. Dabei wird vor allem in Bezug auf die Integration

bemängelt, dass Modelle bisher noch nicht sinnvoll über Werkzeuggrenzen und Abstraktionsebenen hinweg „transportiert“ werden können. An diesen Grenzen besteht noch großes Verbesserungspotential, um den dortigen Verlust von Informationen und das Einbringen von Fehlern zu verhindern. Des Weiteren fehlen den aktuellen Werkzeugen noch einige Modellierungskonzepte, um Automatisierungssysteme umfassend beschreiben zu können. Beispielsweise fehlen häufig noch sinnvolle Möglichkeiten zur Modellierung von hybriden Systemen oder des kontinuierlichen Zeitflusses. Dieses Dokument soll die Arbeitsgruppen im Zentralprojekt auf diese Probleme aufmerksam machen, damit sie sie in ihren weiteren Arbeiten berücksichtigen können.

Literaturverzeichnis

1. Stahl et al. 2005
   Thomas Stahl, Markus Völter. Modellgetriebene Softwareentwicklung. Techniken, Engineering, Management. Dpunkt Verlag, 2005.
2. Alur et al. 1994
   Rajeev Alur, David L. Dill. A Theory of Timed Automata. Theoretical Computer Science Vol. 126, 1994.
3. Marwedl 2005
   Peter Marwedel. Embedded System Design. Springer, Berlin. 2005.