Modellbasiertes Vorgehen bei Echtzeitanforderungen
Von der Spezifikation bis zur Validierung

Arne Noyer[1,3], Joachim Engelhardt[2], Padma Iyenghar[1],
Jürgen Kreyßig[2], Elke Pulvermüller[1], Jonas Diemer[4], Michael Uelschen[5]

[1]Universität Osnabrück, [2]Ostfalia Hochschule,
[3]Willert Software Tools GmbH, [4]Symtavision GmbH,
[5]Hochschule Osnabrück

Beim Embedded Software Engineering gehören Zeitanforderungen zu den wichtigsten nicht-funktionalen Anforderungen. Deshalb gibt es spezialisierte Werkzeuge, die das Zeitverhalten eines realisierten Embedded Software Systems analysieren und validieren können. Unabhängig davon nimmt die modellbasierte Softwareentwicklung weiter an Bedeutung zu, um die steigende Komplexität der Embedded Software zu beherrschen. Etabliert haben sich als Modellierungssprachen u.a. die Unified Modeling Language (UML) und Matlab/Simulink. Es ist auch möglich, Zeitverhalten für Modell-Elemente zu definieren. Darüber hinaus ist es immer noch von Vorteil, Anforderungen in spezialisierten Werkzeugen für Anforderungsmanagement zu verwalten. Es wird ein Workflow zur Integration von Zeitanforderungen von der Spezifikation bis zur Validierung präsentiert. Dieser schließt die Lücken zwischen den unterschiedlichen Entwurfsdomänen.

Einleitung

Eingebettete Systeme werden zunehmend für unterschiedliche Einsatzzwecke verwendet. Dabei müssen diese immer mehr Funktionen übernehmen, die oft auch voneinander abhängen. Dies resultiert in einer steigenden Komplexität für die Softwareentwicklung für solche Systeme. Eine häufig verwendete Herangehensweise zur Beherrschung der Komplexität ist der Einsatz von modellbasierter Softwareentwicklung. Neben anderen Sprachen hat sich hier die Unified Modeling Language (UML) [1] als Standard etabliert. Das Spezifizieren von Anforderungen ist dennoch essenziell.

Für das Anforderungsmanagement werden zumeist spezielle Werkzeuge wie DOORS [2] und Polarion [3] eingesetzt. Anforderungen an das Zeitverhalten können hier ebenfalls bereits textuell und/oder durch selbst definierte Attribute erfasst werden. Um entlang des gesamten Softwareentwicklungsprozesses sicherstellen zu können, dass die Anforderungen berücksichtigt werden, ist die Verfolgbarkeit von Anforderungen essenziell. Nur so kann sichergestellt werden, dass sämtliche Anforderungen umgesetzt werden und bei deren Änderungen analysiert werden, welche zugehörigen Modellelemente angepasst werden müssen (Impact Analyse).

Über das textuelle Erfassen von Zeitanforderungen hinaus können diese jedoch auch in Modellen weiter spezifiziert werden, indem bspw. direkt für eine modellierte Operation hinterlegt wird, wie lang ihre maximale Ausführungszeit sein darf. Darüber hinaus lassen sich über Modelle weitere Aspekte wie Tasks mit ihrer Periode, Priorität und Ausführungszeiten modellieren sowie Operationen den Tasks

zuordnen. Zur Modellierung solcher Eigenschaften kommen hier spezialisierte Werkzeuge zur Analyse von Zeitverhalten wie SymTA/S [4] zum Einsatz, mit denen sich das Zeitverhalten einer Software analysieren und validieren lässt. Zeiteigenschaften können jedoch auch in anderen Modellierungssprachen wie der UML erfasst werden. Damit solche Modelle konsistent mit denen in Analysewerkzeugen sind, ist auch hier die Verfolgbarkeit von Elementen von großer Bedeutung. Auch die Synchronisierung von Daten spielt hier eine wichtige Rolle.

Im Folgenden wird anhand eines Beispiels ein Einsatz vorgestellt, wie Anforderungen an das Zeitverhalten mit UML-Modellen synchronisiert werden, die Zeiteigenschaften in UML weiter beschrieben werden und anschließend mit einem Analysewerkzeug validiert werden.

Verfolgen von Zeitanforderungen

In Abbildung 1 sind Anforderungen an ein Teilsystem für einen Akkuschrauber in einem Anforderungsmanagementwerkzeug dargestellt. Dabei wurden Zeitanforderungen nicht nur textuell erfasst, sondern in dem zusätzlich definierten Attribut *Boundary* konkrete Werte für Zeitschranken beschrieben.

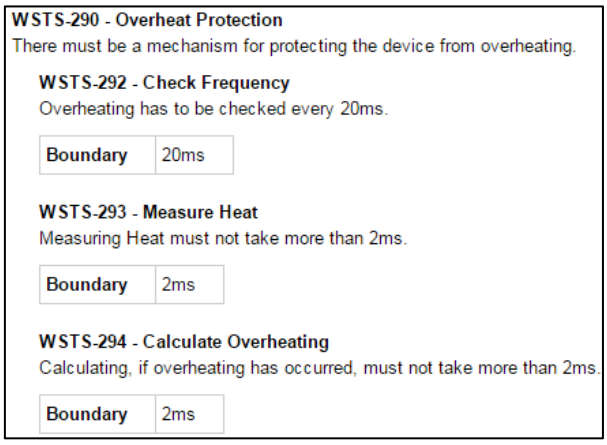

Abbildung 1: Anforderungen an einen Akkuschrauber

Die Software, die die Anforderungen realisiert, wurde modellbasiert mit einem UML-Werkzeug entwickelt. Um Anforderungen in Anforderungsmanagement-werkzeugen mit UML-Elementen zwecks Verfolgbarkeit in Beziehung zu setzen, gibt es verschiedene Ansätze, wie bspw. das Gateway für Rational Rhapsody [5]. Da vorhandene Lösungen oft proprietär für bestimmte Werkzeuge sind, wurden im Projekt die Anforderungen mit dem standardisierten Requirements Interchange Format (ReqIF) [6] zwischen dem Anforderungsmanagement- und dem UML-Werkzeug synchronisiert. Der Ablauf zum Austausch der Anforderungen wird in Abbildung 2 visualisiert.

Im Anforderungsmanagementwerkzeug wurden sowohl ein Lastenheft als auch ein Pflichtenheft erfasst. Das Pflichtenheft wurde anschließend nach ReqIF exportiert und daraufhin mittels eines dafür eigens entwickelten Mechanismus in das UML-Format konvertiert und importiert. Beim Import in UML wird für jedes Anforderungsdokument aus ReqIF jeweils ein UML-Package erstellt, welches die im Dokument enthaltenen Anforderungen enthält. Die verschiedenen Arten dieser

Anforderungen werden über UML-Stereotypen dargestellt, die Inhalte/Eigenschaften über Tagged Values erfasst (siehe Abbildung 3). Diese *Repräsentation* der Anforderungen aus ReqIF in UML ermöglicht nun, dass Beziehungen zwischen UML-Elementen und Anforderungen erstellt werden können. Zum Erstellen von Beziehungen werden im UML-Werkzeug UML-Dependencys verwendet.

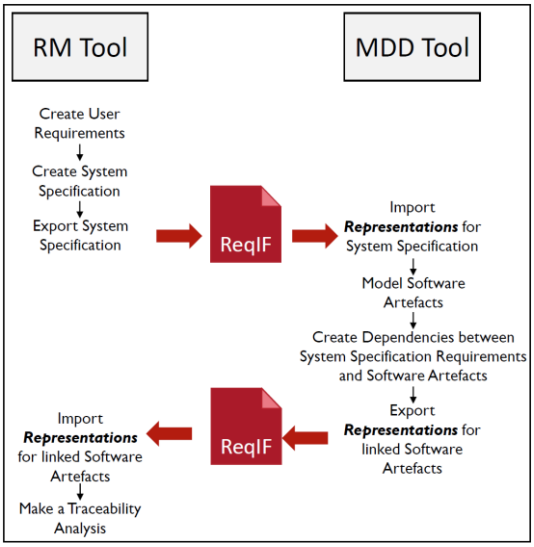

Abbildung 2: Synchronisieren von Anforderungen

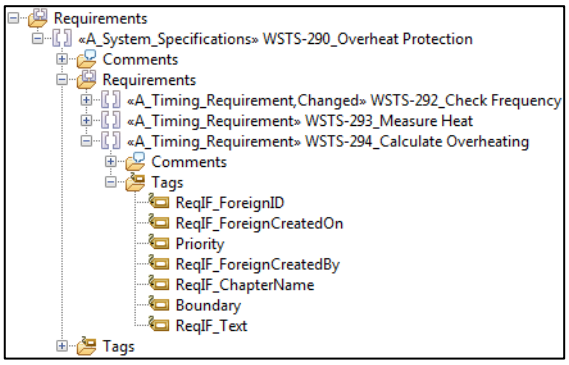

Abbildung 3: Anforderungen in einem UML-Werkzeug

Abbildung 4 zeigt diese Beziehungen zwischen UML-Elementen und Anforderungen. Darüber hinaus sind hier schon Zeiteigenschaften weiter spezifiziert worden, was durch verschiedene Stereotypen und Tagged Values zu sehen ist.

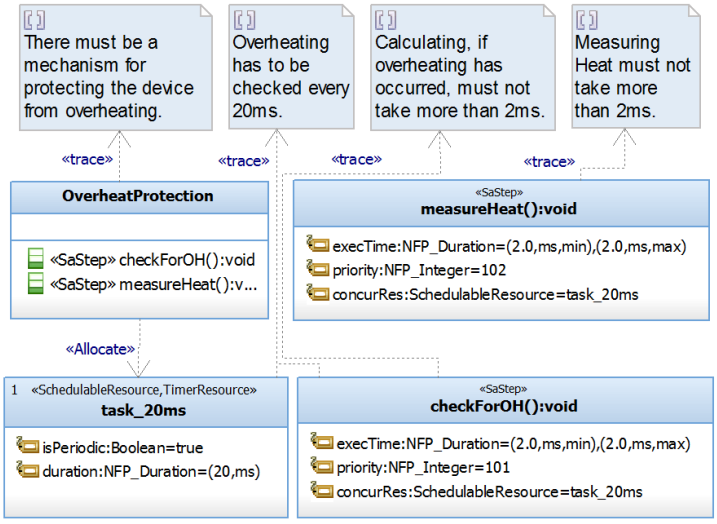

Abbildung 4: Beziehungen zwischen UML-Elementen und Anforderungen im UML-Werkzeug

Um die Beziehungen auch in Anforderungsmanagementwerkzeugen sichtbar zu machen, wurde ein Verfahren entwickelt, dass diese Beziehungen im UML-Modell analysiert die zu einer Anforderung in Beziehung gesetzten UML-Elemente zurück ins ReqIF überführt. Die ReqIF-Datei kann anschließend zurück ins

Anforderungsmanagementwerkzeug importiert werden. Abbildung 5 zeigt, wie Repräsentationen für UML-Elemente und deren Beziehungen im Anforderungsmanagementwerkzeug dargestellt werden. Dieser Prozess ermöglicht eine vollständige Verfolgbarkeit zwischen Anforderungen und UML-Elementen in beiden Richtungen.

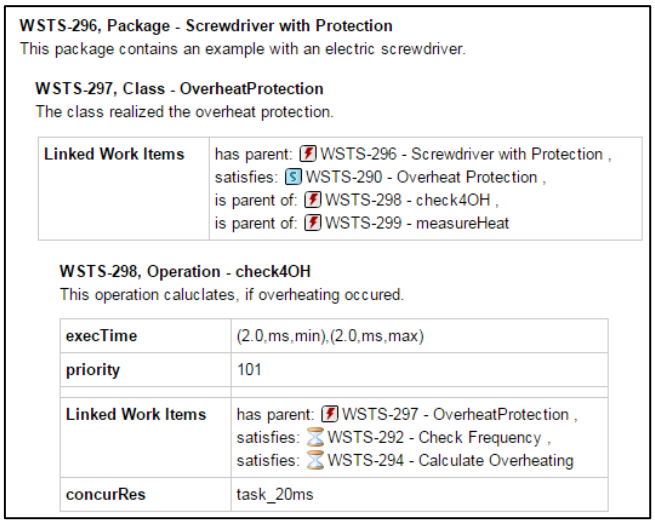

Abbildung 5: Beziehungen zwischen UML-Elementen und Anforderungen im Anforderungsmanagementwerkzeug

Modellbasiertes Spezifizieren von Zeitverhalten

In Abbildung 4 ist bereits zu sehen, dass das Zeitverhalten in UML näher spezifiziert wurde. Hierzu wurde das UML-Profil zum *Modeling and Analysis of Realtime and Embedded Systems (MARTE)* [7] verwendet. Durch den Stereotyp *SaStep* des Profils werden u.a. Ausführungszeiten für Operationen erfasst. Darüber hinaus wurden Details für die Ziel-Plattform modelliert, wie in Abbildung 6 zu sehen ist. Die Ziel-Plattform enthält hier drei Tasks (Stereotyp: SchedulableResource), die mit unterschiedlicher Periodizität ausgeführt werden und verschiedene Ausführungszeiten besitzen. Die Operationen sind diesen Tasks zugeordnet und die Tasks wiederum sind einem CPU Core (Stereotyp: SaExecHost) zugeordnet, der ebenfalls modelliert wurde und für den ein Scheduling Mechanismus definiert wurde.

Zusätzlich zur Plattform wurden mit UML/MARTE ebenfalls Ausführungspfade modelliert, wie in Abbildung 7 dargestellt ist. Der Ausführungspfad gibt an, welche Operationen (Runnables) in welcher Reihenfolge in einem Szenario ausgeführt werden, für das später das Zeitverhalten geprüft werden soll. Außerdem sind hier ebenfalls die Ausführungszeiten für die Operationen zu sehen. Über den Stereotyp *SaEndToEndFlow* wurde die maximal erlaubte Ausführungszeit für den gesamten Pfad definiert.

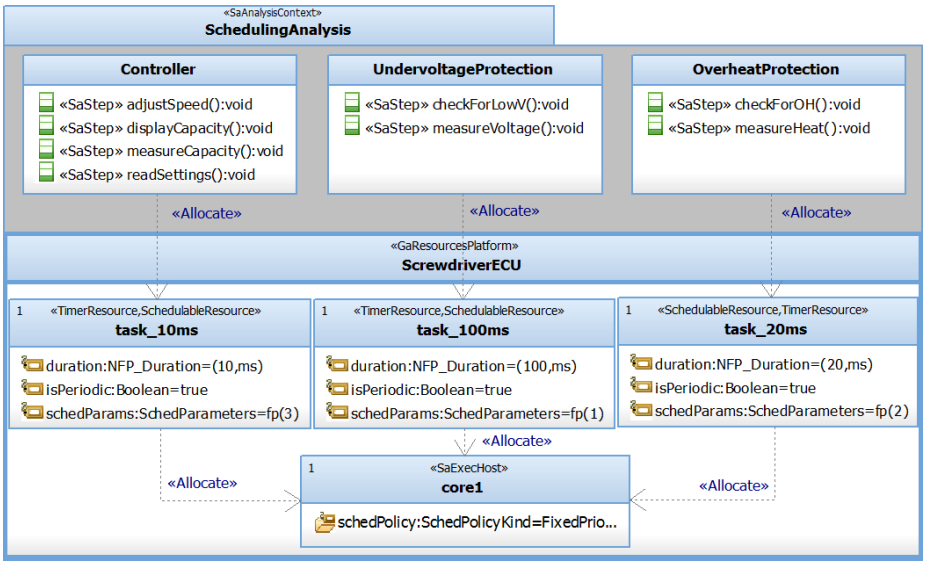

Abbildung 6: Mapping von Tasks mit UML/MARTE

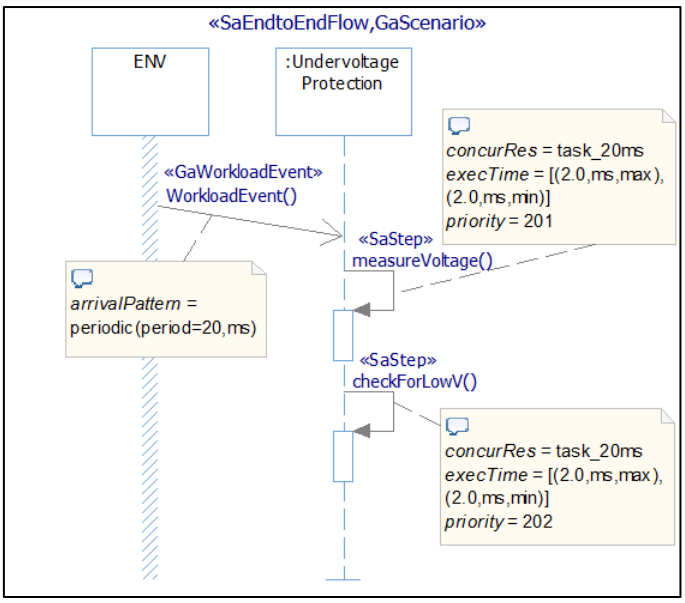

Abbildung 7: Ein Ausführungspfad mit UML/MARTE

Validierung von Zeitanforderungen

Um das modellierte Zeitverhalten analysieren zu können, wurden Modelltransformationen zwischen dem UML-Format und dem Analysewerkzeug SymTA/S [4] durchgeführt. Die Vorgehensweise hierzu zeigt Abbildung 8. Dabei ist es von Vorteil, wenn auch Analyseergebnisse zurück ins UML-Modell übertragen werden, damit ein UML-Entwickler dort direkt an betreffenden Elementen überprüfen kann, ob das Zeitverhalten gültig ist.

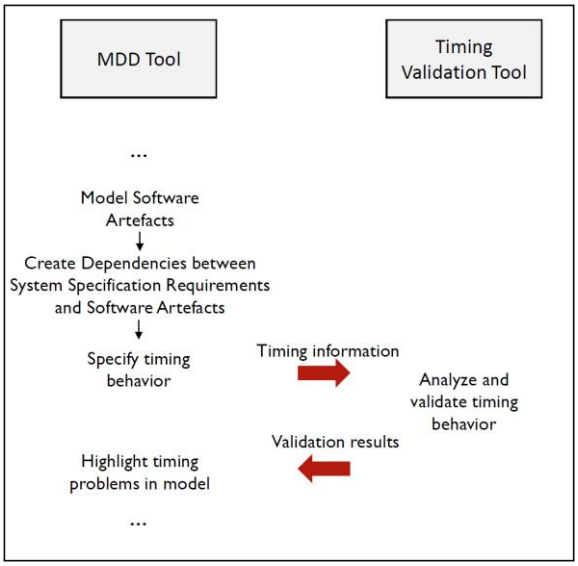

Abbildung 8: Ablauf zum Validieren der Zeiteigenschaften

Neben weiteren Analyse-Methoden unterstützt SymTA/S die Durchführung von Scheduling-Analysen. Das Ergebnis solch einer Analyse kann zur Übersicht wie in Abbildung 9 visualisiert werden.

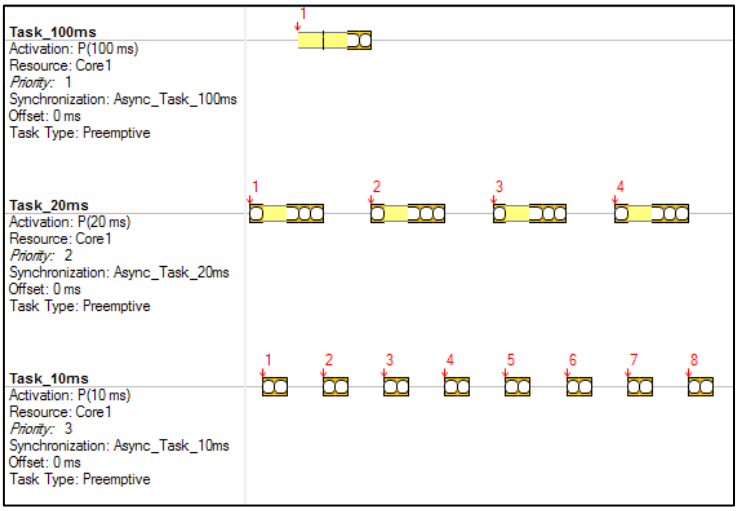

Abbildung 9: Scheduling von Tasks

Ein Tortendiagramm zeigt die Auslastung einer CPU (siehe Abbildung 10). Im Beispiel beträgt die Gesamt-Auslastung der CPU 84%, sodass die Zeiteigenschaften von den modellierten Tasks eingehalten werden können. Das Analyseergebnis kann in UML erneut über Stereotypen und Tagged Values sichtbar gemacht werden. Beim modellierten Core im UML-Modell wurde der hierfür speziell geeignete Stereotyp *SaExecHost* verwendet.

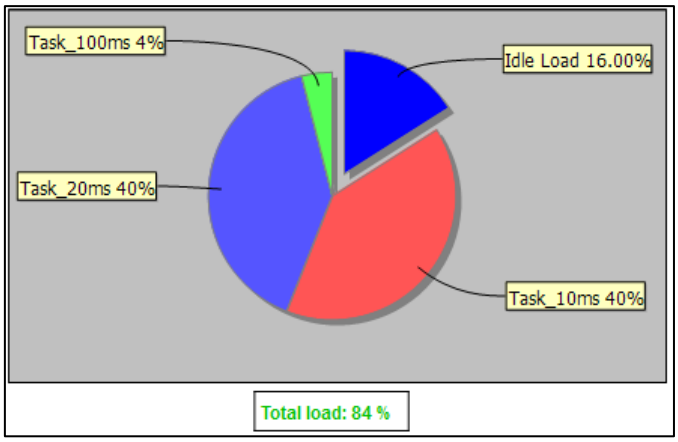

Abbildung 10: Diagramm zur Prozessorauslastung

Synchronisierung von Zeiteigenschaften zwischen unterschiedlichen Modellierungsdomänen

Neben der UML gibt es weitere Modellierungssprachen, die zur Systembeschreibung eingesetzt werden. Gerade im Bereich der eingebetteten Systeme ist Matlab/Simulink [8] weit verbreitet. Zeitanforderungen lassen sich hier jedoch nicht unmittelbar eingeben. Für eine Machbarkeitsstudie zum Umgang dieses Problems wurde zunächst eine einfache Zeitanalyse eines Matlab/Simulink-Modells durch das Zeitanalyse-Werkzeug SymTA/S angestrebt. Dazu sollten *Runnables* und *Tasks* in Simulink angegeben werden können sowie *Periode* und *Core Execution Time (CET)* einer Runnable als Zeitanforderungen für eine Analyse. Zur Eingabe dieser Daten in das Simulink-Modell wurde eine benutzerdefinierte Maske mit den Werten *Period* und *CET(min, max)* erstellt, die auf Matlab Sub-System-Blöcke angewendet werden kann.

Abbildung 11 zeigt die Steuerung eines Akkuschraubers. Operationen werden in Sub-System-Blöcke gruppiert, die hier als *runnable 1* und *runnable 2* zu sehen sind. Für die Zeitanalyse werden diese Blöcke dann als *Runnable* interpretiert, die darin enthaltenen Operationen als *Tasks*. Mit der Eingabemaske können *Period* und *CET* für das jeweilige Runnable spezifiziert werden.

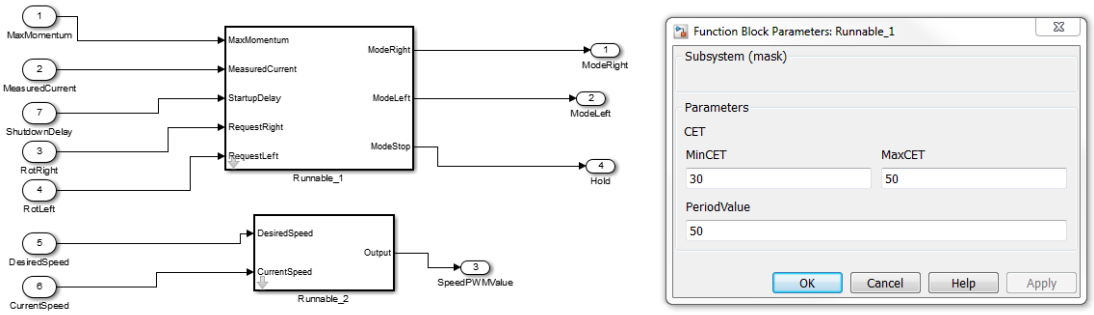

Abbildung 11 Zeitanforderungen in Matlab Simulink

Zeiteigenschaften können also in ReqIf, UML und Matlab Simulink hinterlegt werden. In realen Projekten sind diese Informationen tatsächlich oft in mehreren

Modellen vorzufinden. Um die Verfolgbarkeit in jeder Domäne für sich zu gewährleisten, erfordern die bisher vorgestellten Mechanismen redundante Informationen. Automatische Modelltransformationen erleichtern dabei zwar die Übertragung der Information zwischen den Domänen, es liegt jedoch das Problem nahe, dass Änderungen an diesen Informationen in andere Domänen propagiert werden müssen.

Derzeit wird in einem Forschungsprojekt ein Verfahren entwickelt, wie Zeiteigenschaften über verschiedene Domänen hinweg synchronisiert, verfeinert und bereits auf Modellebene validiert werden können. Dazu werden die vorgestellten Mechanismen zur Verfolgbarkeit genutzt und erweitert, um die Datenwerte werkzeugübergreifend korrekt abzugleichen. Wie am Beispiel von Simulink zu sehen ist, kann nicht jede Domäne die Verfolgbarkeits-Beziehungen direkt erfassen. Dazu soll ein zentrales Repository die Abhängigkeiten zwischen den verschiedenen Zeiteigenschaften speichern. Über definierte Schnittstellen sollen dann werkzeugspezifische Erweiterungen diese Informationen sowie weitere Hinweise wie Analyseergebnisse aus diesem Repository abrufen können.

Zusammenfassung

Es wurde anhand eines Beispiel-Projektes gezeigt, wie (Zeit-)Anforderungen über das standardisierte Requirements Interchange Format (ReqIF) zwischen Anforderungsmanagementwerkzeugen und UML synchronisiert werden können. Anschließend wurde das Zeitverhalten mittels dem MARTE-Profil in UML näher beschrieben. Informationen aus dem UML-Modell wurden daraufhin per Modelltransformation in ein Analysewerkzeug überführt, sodass dort Analysen für das Zeitverhalten durchgeführt werden konnten. Der Prozess ermöglicht, dass frühzeitig während der Entwicklung Zeiteigenschaften beschrieben und validiert werden können. Ein iteratives Anwenden des Ablaufes ist dabei empfehlenswert.

Es wurde gezeigt, dass die Anbindung weiterer Werkzeuge teilweise aufwändiger ist. Ein aktuelles Forschungsprojekt zielt auf noch engere Kollaboration von verschiedenen Domänen bei der Bewältigung von komplexen Zeitanforderungen.

Literaturverzeichnis

[1] Object Management Group, „UML Spezifikation 2.5," 2015. [Online]. http://www.omg.org/spec/UML/

[2] IBM, „IBM Rational DOORS," 2014. [Online]. http://www-03.ibm.com/software/products/de/ratidoor

[3] Polarion Software, „Polarion," 2014. [Online]. https://www.polarion.com/products/alm/index.php

[4] Symtavision GmbH, „SymTA/S and Traceanalyzer," [Online]. https://www.symtavision.com/products/symtas-traceanalyzer/

[5] IBM, „Managing requirements with Rhapsody Gateway and DOORS,“ [Online] http://pic.dhe.ibm.com/infocenter/rhaphlp/v7r6/index.jsp?topic=%2Fcom.ibm.rhp.doors.tutorial.doc%2Ftopics%2Fabstract_rhpdoors.html

[6] Object Management Group, „Requirements Interchange Format (ReqIF),“ 2013. [Online]. http://www.omg.org/spec/ReqIF/

[7] Object Management Group, „UML Profile For MARTE: Modeling And Analysis Of Real-Time Embedded Systems,“ 2011, [Online]. http://www.omg.org/spec/MARTE/

[8] MathWorks, „Matlab/Simulink,“ 2015. [Online]. http://de.mathworks.com/products/simulink/

Angaben zu den Referenten:

Arne Noyer
anoyer@uni-osnabrueck.de

Arne Noyer hat bereits viele Erfahrungen mit Vorträgen auf Konferenzen in Industrie und Wissenschaft und ist in der Forschung und Entwicklung tätig. Aktuell liegt sein Forschungsschwerpunkt in einem Projekt, das im Bereich der modellbasierten Beschreibung von Zeitanforderungen und dessen Validierung ist. Weiterhin hat er langjährige Erfahrungen in den Disziplin UML, Codegenerierung, Eclipse-Plugin-Entwicklung und Lehre/Schulungen.

Joachim Engelhardt
jo.engelhardt@ostfalia.de

Joachim Engelhardt ist wissenschaftlicher Mitarbeiter an der Ostfalia Hochschule für angewandte Wissenschaften. Aktuell führt er eine Lehrveranstaltung im Bereich der modellbasierten Softwareentwicklung für eingebettete Systeme durch. Außerdem arbeitet er in verschiedenen Forschungsprojekten. Derzeit liegt der Fokus seiner Forschungsaktivitäten im Bereich des modellbasierten Erfassens von Zeiteigenschaften.