Modellgetriebene Validierung von System-Architekturen

André Pflüger, Wolfgang Golubski

Westsächsische Hochschule Zwickau
Fachgruppe Informatik
Dr.-Friedrichs-Ring 2A
08056 Zwickau
{Andre.Pflueger|golubski}@fh-zwickau.de

Stefan Queins

SOPHIST GmbH
Vordere Cramergasse 13
90478 Nürnberg

Stefan.Queins@sophist.de

Abstract: Die System-Architektur bildet das Fundament eines jeden eingebetteten Systems. Der Architekt muss sie den sich ständig ändernden Bedingungen anpassen und die Validität gegenüber den System-Anforderungen nachweisen. Der vorgestellte Ansatz reduziert diese zeitaufwändige Arbeit durch einen automatisierbaren Validierungsprozess auf Basis der Unified Modeling Language (UML) und unterstützt den Architekten beim Architekturentwurf und der Impact-Analyse.

1. Motivation

Bei softwareintensiven, eingebetteten Systemen ist eine Projektlaufzeit über mehrere Jahre eher die Regel als eine Ausnahme. Die Entwicklung wird mit sich häufig ändernden Bedingungen konfrontiert, die eine kontinuierliche Planung und Entwicklung auf Basis vereinbarter Anforderungen fast unmöglich machen. Der Kunde verlangt vom Hersteller größtmögliche Flexibilität bei der Systementwicklung. Gleichzeitig steigt die Komplexität der zu entwickelnden Systeme kontinuierlich und damit auch die Anzahl und Beziehungen der Anforderungen untereinander. Mittlere bis hohe vierstellige Werte sind nicht ungewöhnlich.

Das Fundament eingebetteter Systeme bildet die System-Architektur. Sie wird beispielsweise durch die Faktoren

- funktionale und nicht-funktionale Anforderungen,
- eingesetzte Technik und Technologien,
- Verständnis des aktuellen Systems,
- Erfahrungen aus früheren Entwicklungen sowie
- dem Faktor Mensch

beeinflusst und ist für die Abschätzung der Entwicklungskosten und -zeit von Bedeutung. Es sind die Aufgaben des Architekten sie unter Berücksichtigung aller Einflussfaktoren zu entwerfen und deren Validität gegenüber den architekturrelevanten Anforderungen sicherzustellen. Diese Validierung ist ein arbeitsintensiver und zeitaufwendiger Vorgang, der bei jeder Änderung eines Einflussfaktors, z.B. einer Anforderung, wiederholt werden muss. Dabei sieht das Vorgehen des Architekten in vielen Fällen folgendermaßen aus:

Der Architekt erstellt einen ersten Architekturentwurf und verfeinert diesen iterativ. Für die Validierung wird die Methode des scharfen Hinsehens verwendet. Damit ist ein manueller Abgleich der Architektur mit den Anforderungen ohne ein definiertes Vorgehen gemeint, was zum unabsichtlichen Auslassen einiger Aspekte führen kann. Die Validierung beruht auf Abschätzungen, die auf Erfahrungen des Architekten basieren, und nur selten, meist bei kritischen Anforderungen, auf belegbaren Zahlen. Aufgrund des Zeit- und Arbeitsaufwands wird die Dokumentation und Validierung nur für bestimmte, beispielsweise Review-Termine vollständig durchgeführt.

Der in diesem Paper vorgestellte Ansatz definiert einen teilautomatisierbaren Validierungsprozess, der in das iterative Vorgehen zum Entwurf der System-Architektur eingebaut wird. Sogenannte Validierungsziele fassen architekturrelevante Anforderungen auf einem für den Architekturentwurf passenden Abstraktionsniveau zusammen, was den Überblick vereinfacht und die komplexen Anforderungsbeziehungen untereinander sichtbar werden lässt. Zu jedem Validierungsziel legt der Architekt ein Verfahren zur Feststellung der Validität eines Validierungsziels fest, das sogenannte Validierungsverfahren. Die Validierung der System-Architektur ist erfolgreich, falls alle Validierungsziele erfolgreich validiert werden. Der Ansatz setzt eine modellbasierte Dokumentation des Systems mit UML voraus. Für jedes Validierungsziel wird eine validierungszielspezifische Notation erstellt, mit der die für das zugehörige Validierungsverfahren benötigten Informationen im sogenannten Validierungsmodell modelliert werden. Die Modelldaten werden zum einen für die Validierung und zum anderen für eine lückenlose Dokumentation des Architektur-Entwurfs, dessen Entstehung und Anpassung genutzt. Änderungen zum Finden einer passenden System-Architektur werden direkt am Modell vorgenommen. Der Ansatz reduziert den Arbeits- und Zeitaufwand für die Validierung, indem möglichst viele Validierungsverfahren automatisiert durchgeführt und die dazugehörigen Werte direkt aus dem Validierungsmodell ausgelesen werden. Der Architekt wird beim Entwurf und der Impact-Analyse nach Anforderungsänderungen unterstützt und Änderungen an der Architektur können aufgrund der lückenlosen modellbasierten Dokumentation nachvollzogen werden.

Im weiteren Verlauf des Papers wird in Kapitel 2 zunächst auf verwandte Arbeiten verwiesen. Danach folgt in Kapitel 3 die Beschreibung eines Beispiels aus dem Radar-Bereich, das bei der Beschreibung des Validierungsprozesses in Kapitel 4 zur Veranschaulichung herangezogen wird. Kapitel 5 listet die Vorteile und die bisherigen Erfahrungen mit dem Ansatz auf, bevor Kapitel 6 ein Fazit zieht und einen Ausblick über zukünftig geplante Entwicklungen gibt.

2. Verwandte Arbeiten

Bei unseren Nachforschungen haben wir keinen Ansatz entdeckt, der einen modellgetriebenen Validierungsprozess für die System-Architektur gegenüber den System-Anforderungen definiert und den Architekten bei der Entwurfsarbeit und der Impact-Analyse unterstützt.

Das Tool PREEvision von aquintos [Aqu11] erlaubt die Modellierung, Entwicklung und Bewertung von Elektronik/Elektrik (E/E)-Architekturen. Für die Modellierung steht eine auf einem eigenen Metamodell basierende domänenspezifische grafische Notation und ein Datenmodell zur Eingabe der architekturrelevanten Informationen zur Verfügung. Der Hauptfokus liegt auf der Entwicklung und Validierung der E/E-Architektur, während sich unser Ansatz auf die Validierung der System-Architektur als Ganzes, Kombination von Soft- und Hardware, konzentriert. Statt eines eigenen domänenspezifischen Metamodells setzt unser Ansatz auf eine Erweiterung der domänenspezifischen Notation mit Hilfe des Profilmechanismus der UML, wodurch die Erweiterung vorhandener domänenspezifischer Notationen für die Validierung mit geringem Aufwand und validierungszielspezifisch möglich ist.

Für die Modellierung von System-Architekturen kann die System Modeling Language (SysML) [OMG10a], die Architecture Analysis & Design Language (AADL) [CMU11] oder das für eingebettete und Echtzeit-Systeme gedachte Modeling and Analysis of Real-Time and Embedded Systems (MARTE)-Profil [OMG10b] eingesetzt werden. Für die Erstellung von speziell auf die einzelnen Validierungsziele angepassten Notationen sind sie allerdings ungeeignet.

Das vom Aerospace Vehicle Systems Institute initiierte Projekt System Architecture Virtual Integration (SAVI) [Fei10] umfasst drei Schlüsselkonzepte: ein Referenzmodell, ein Modell-Repository mit -Bus sowie ein modellbasierter Entwicklungsprozess. Das Konzept ermöglicht auf das Referenzmodell abgebildete Modelle, z.B. auf Basis von UML, AADL oder SysML, mit in einem Repository hinterlegten Anforderungen zu vergleichen. Damit ließe sich zwar theoretisch eine Validierung der System-Architektur gegenüber Anforderungen durchführen, der Mehraufwand wäre aber sehr groß, da das Projekt eine deutlich andere Zielsetzung hat. Zudem existiert es bisher nur in der Beschreibung der Machbarkeitsstudie.

Als ein Beispiel für Ansätze zur Verifikation von System-Architekturen wird CoFluent Studio [CoF10] vorgestellt. Es bietet ein Werkzeug zum Testen der System-Architektur in einer virtuellen Umgebung an. Für die Dokumentation werden UML-Modelle inklusive der Profile SysML und MARTE eingesetzt. Diese Modelle werden in ausführbare SystemC-Transaction-Level-Modelle transformiert. Das Konzept ist vergleichbar zum Executable UML [Bal02], hat aber eine andere Zielsetzung gegenüber unserem Ansatz. CoFluent Studio verifiziert die modellierten Informationen durch eine Simulation in einer speziellen Umgebung. CoFluent Studio könnte als Validierungsfahren zum Testen eines Validierungsziels verwendet werden, es ermöglicht aber weder die Validierung der System-Architektur als Ganzes, noch unterstützt es den Architekten beim Entwurf oder der Impact-Analyse.

3. Beispiel für die Validierung einer System-Architektur

Die Beschreibung des Validierungsprozesses und der Begrifflichkeiten im folgenden Kapitel werden mit Hilfe eines stark vereinfachten Radar-Systems unterstützt. Die Funktionsweise wird anhand der in Abbildung 1 dargestellten Software-Sicht erläutert. Sie ist Teil der System-Architektur-Dokumentation im Entwicklungsmodell.

Die Reflektionen der vom Radar ausgestrahlten elektromagnetischen Wellen (Signale) werden über die Antenne empfangen und in vier Schritten zu den für die Anzeige auf dem Radarbildschirm benötigten Tracks verarbeitet. Während der Signalverarbeitung werden die empfangenen Reflektionen zunächst digitalisiert und zu sogenannten Beams zusammengefasst. Aus diesen werden Störungen herausgefiltert, so dass nur noch die von Objekten reflektierten Signale, sogenannte Hits, übrig bleiben. In der nächsten Verarbeitungsstufe, der Plotverarbeitung, werden alle zu einem Objekt gehörenden Hits zu Plots zusammengefasst bevor sie in der Trackverarbeitung mit weiteren z.B. historischen Daten verknüpft werden, um beispielsweise die Bewegung eines Objektes auf dem Radarbildschirm verfolgen zu können.

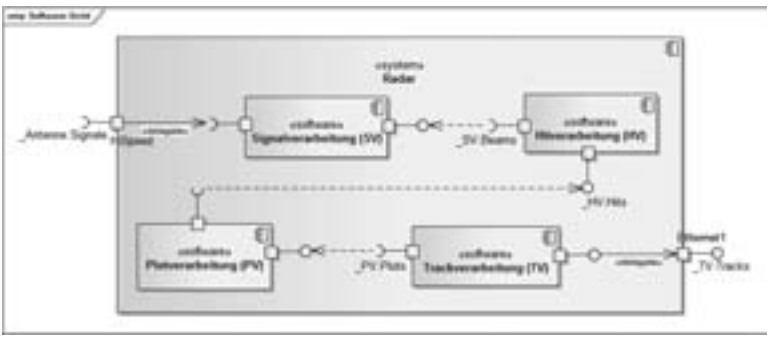

Abbildung 1: Software-Sicht des Radar-Systems im Entwicklungsmodell

Bei der Radarverarbeitung müssen große Mengen an Daten kontinuierlich und in Echtzeit verarbeitet werden. Die früher fast nur aus Hardware bestehenden Systeme werden heutzutage meist durch eingebettete Systeme realisiert. In der System-Architektur muss festgelegt werden auf welcher Verarbeitungseinheit die Software-Komponenten ausgeführt werden, welche logischen Schnittstellen diese haben und über welche physikalischen Schnittstellen diese übertragen werden. Ein im Entwicklungsmodell dokumentierter Entwurf des Architekten ist in Abbildung 2 zu sehen.

Die zwei folgenden Anforderungen an das Radar-System werden in Kapitel 4 zur Verdeutlichung des Validierungsprozesses und den dazugehörigen Begriffe verwendet:

- Das System muss eingespeiste Testsignale innerhalb von 220ms verarbeiten. (A1)
- Die durchschnittliche Prozessorauslastung muss kleiner gleich 60% sein. (A2)

Das Beispiel dient der Verdeutlichung des Ansatzes, weshalb wir viele einflussnehmende Aspekte, z.B. Latenzzeiten durch Kommunikation, sowie die Beeinflussung der Signale durch die kontinuierliche Verarbeitung an dieser Stelle unberücksichtigt lassen.

4. Der Validierungsprozess

Das Hauptziel beim Entwurf der System-Architektur ist, den Anforderungen des Kunden gerecht zu werden. Dies kann durch die Validierung der Architektur gegenüber den architekturrelevanten Anforderungen sichergestellt werden. Abbildung 3 stellt in einem Aktivitätsdiagramm die dafür benötigten Schritte unseres Validierungsprozesses dar.

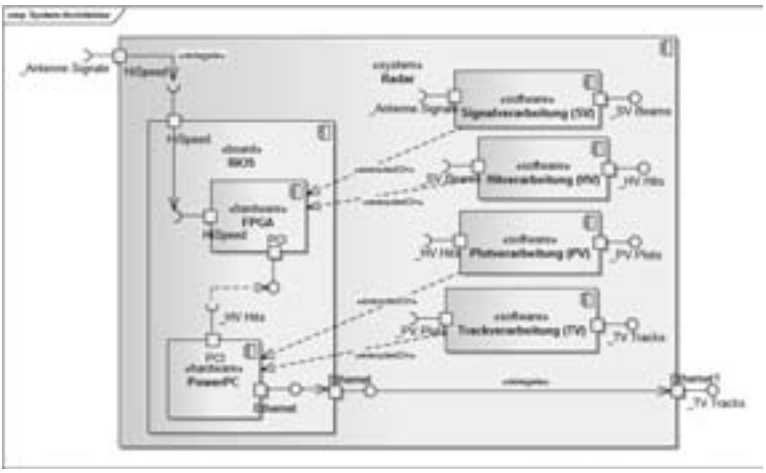

Abbildung 2: Zuordnung der Software-Komponenten zu den Verarbeitungseinheiten des Radar-Systems inklusive der physikalischen und logischen Schnittstellen

Aus der Menge der architekturrelevanten Anforderungen ermittelt der Architekt zunächst die Validierungsziele. Ein Validierungsziel beschreibt einen architekturspezifischen Aspekt auf einem für die System-Architektur passenden Abstraktionslevel. Dem Validierungsziel ist mindestens eine Anforderung zugeordnet. Zu jedem Validierungsziel gehört ein vom Architekten festgelegtes Validierungsverfahren, mit dem das Validierungsziel gegenüber den zugeordneten Anforderungen getestet werden kann. Im vorgestellten Radar-Beispiel aus Kapitel 3 ermittelt der Architekt aus den beschriebenen Beispielanforderungen A1 und A2 die Validierungsziele

- „Gesamtverarbeitungszeit“ (VZ1) mit der zugeordneten Anforderung A1 und
- „Durchschnittliche Prozessorauslastung“ (VZ2) mit der zugeordneten Anforderung A2.

Im nächsten Schritt erarbeitet der Architekt ein Validierungsmodell. Für jedes Validierungsziel legt der Architekt ein Validierungsverfahren fest und erstellt eine validierungszielspezifische Notation, in der er definiert, wie die für das Validierungsverfahren benötigten Informationen im Validierungsmodell zu modellieren sind. Die Eingabe der Daten kann durch die Übernahme vorhandener Daten aus dem Entwicklungsmodell und durch eine manuelle Eingabe erfolgen. Das fertige Validierungsmodell enthält also die für die Durchführung aller Validierungsverfahren benötigten Informationen auf Basis der jeweiligen validierungszielspezifischen Notationen.

Die Erstellung des Validierungsmodells soll zunächst am Radar-Beispiel verdeutlicht werden. Der Architekt muss jeweils ein Validierungsverfahren und eine validierungszielspezifischen Notation für die Validierungsziele VZ1 und VZ2 festlegen und die dazugehörigen Informationen im Validierungsmodell modellieren. Zur Validierung von VZ1 muss die Gesamtverarbeitungszeit des Systems ermittelt werden, die sich in diesem Beispiel aus der Summe der Verarbeitungszeiten aller Software-Komponenten i ergibt (siehe Formel 1).

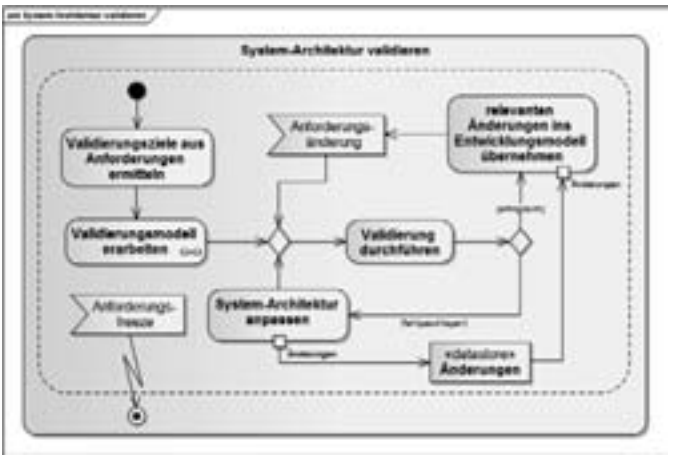

Abbildung 3: Ablauf der Validierung der System-Architektur

Formel 1: Gesamtverarbeitungszeit

$$executionTime_{total} = \sum_{i=1}^{n} executionTime_i$$

Formel 2: durchschnittliche Prozessorauslastung

$$processorLoad_p = \sum_{i=1}^{n} \frac{executionTime_i}{n}$$

Die Validierung ist erfolgreich, falls der Wert kleiner gleich des in A1 festgelegten Wertes (220ms) ist, ansonsten schlägt sie fehl. Es wird angenommen, dass dem Architekten die Verarbeitungszeiten der Software-Komponenten auf den verwendeten Verarbeitungsbausteinen z.B. aus früheren Projekten bekannt sind. Für dieses Verfahren werden die Software-Komponenten, die Verarbeitungsbausteine, die Zuordnung der Software-Komponenten zu den Verarbeitungsbausteinen sowie der Verarbeitungszeitwert der einzelnen Software-Komponenten benötigt. Bis auf die Verarbeitungszeitwerte sind alle Informationen bereits im Entwicklungsmodell vorhanden (siehe Abbildung 2), so dass der Architekt die verwendete Notation der benötigten UML-Elemente in die validierungsspezifische Notation übernimmt. Vervollständigt wird die validierungszielspezifische Notation von VZ1 durch ein UML-Profil (siehe Abbildung 4), welches den Software-Komponenten einen Tagged-Value *executionTime* zur Modellierung der Verarbeitungszeitwerte hinzufügt. Auf Basis dieser Notation erstellt der Architekt im Validierungsmodell das in Abbildung 5 dargestellte Komponentendiagramm.

Für das Validierungsziel VZ2 muss die durchschnittliche Prozessorauslastung einer jeden Verarbeitungseinheit p ermittelt werden. Sie ergibt sich vereinfacht aus der Summe der Verarbeitungszeiten der Software-Komponenten i geteilt durch die Anzahl aller auf p verarbeiteten Software-Komponenten n (siehe Formel 2). Die Validierung ist erfolgreich, falls der Wert kleiner gleich des in A2 festgelegten Wertes (60%) ist, ansonsten schlägt sie fehl. Für VZ2 kann die validierungsspezifischen Notationen von VZ1 wiederverwendet werden. Durch sie sind bereits alle benötigten Informationen im Validierungsmodell. Nach der Erarbeitung des Validierungsmodells durch den Architekten folgt im nächsten Schritt die Validierung der System-Architektur (siehe Abbildung 3). Diese setzt sich aus der Validierung aller Validierungsziele zusammen.

Schlägt eine der Validierungen fehl, muss der Architekt die System-Architektur anpassen und die Validierung erneut durchführen. Die für das Entwicklungsmodell relevanten Änderungen an der System-Architektur, z.B. Änderung der Zuordnung einer Software-Komponente zu einer Verarbeitungseinheit, werden dabei zwischengespeichert und im Falle einer erfolgreichen Validierung in das Entwicklungsmodell übernommen.

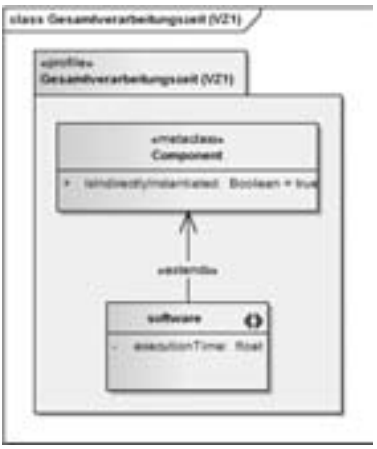

Abbildung 4: UML-Profil für VZ1 (Die Darstellung ergibt sich durch das verwendete Werkzeug Enterprise Architect [Spa10] und ist nicht UML-konform)

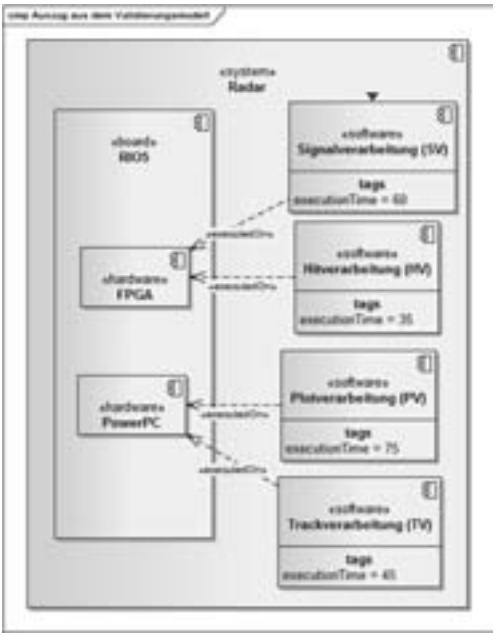

Abbildung 5: Auszug aus dem Validierungsmodell

Zum Abschluss dieses Kapitels werden nun die Validierungsverfahren von VZ1 und VZ2 mit konkreten Verarbeitungszeiten für die vier Software-Komponenten des Radar-Systems durchgeführt und danach die Auswirkungen einer Anforderungsänderung gezeigt. Für die Verarbeitungswerte werden die in Abbildung 5 modellierten Werte verwendet. Die Ergebnisse der Validierung beider Validierungsziele sind in Tabelle 1 und 2 zu sehen. Die Validierung der System-Architektur ist erfolgreich, da die Gesamtverarbeitungsdauer von 215ms kleiner als die in A1 festgelegten 220ms ist und die durchschnittliche Prozessorauslastung des FPGA (44%) und des PowerPCs (56%) je kleiner als die in A2 festgelegten 60% ist. Eine Veränderung der maximalen Gesamtverarbeitungsdauer für die Testsignale in Anforderung A1 auf 170ms hat folgende Auswirkungen: Die Veränderung des Vergleichswertes bewirkt keine Veränderungen an den Validierungszielen und den dazugehörigen Validierungsverfahren. Nach dem in Abbildung 3 beschriebenen Validierungsprozesses erfolgt nach einer Anforderungsänderung die erneute Validierung der System-Architektur, in unserem Beispiel also der Validierungsziele VZ1 und VZ2. Die Validierung der System-Architektur schlägt fehl, da die Validierung von VZ1 fehlschlägt, 215ms ist größer als 170ms. Der Architekt ordnet daraufhin die Software-Komponenten „Plotverarbeitung“ dem FPGA zu und führt die Validierung erneut aus. Die Ergebnisse sind in Tabelle 3 und 4 zu sehen.

Tabelle 1: Ergebnis des Validierungsverfahrens von VZ1

Software-Komponente	Verarbeitungseinheit	Verarbeitungszeit [ms]
Signalverarbeitung	FPGA	60
Hitverarbeitung	FPGA	35
Plotverarbeitung	PowerPC	75
Trackverarbeitung	PowerPC	45
Gesamtverarbeitungszeit [ms]		**215**

Tabelle 2: Ergebnis des Validierungsverfahrens von VZ2

Verarbeitungseinheit	durchschnittliche Prozessorauslastung [%]
FPGA	44
PowerPC	56

Durch die Änderung konnte zwar VZ1 erfolgreich validiert werden, 165ms ist kleiner gleich 170ms, jedoch hat die Änderung der Zuordnung Auswirkungen auf VZ2. Die Validierung der System-Architektur schlägt erneut fehl. Die Validierung schlägt fehl, da 73% größer den in A2 geforderten 60% ist. Das Problem kann beispielsweise durch die Aufteilung der „Plotverarbeitung“ in zwei Software-Komponenten gelöst werden. Das Beispiel zeigt, dass durch die erneute Validierung aller Validierungsziele auch die nicht direkt für den Architekten ersichtlichen Auswirkungen seiner Änderung erkannt werden.

Tabelle 3: Ergebnis des Validierungsverfahrens von VZ1 nach der Anforderungs- und System-Architektur-Änderung

Software-Komponente	Verarbeitungseinheit	Verarbeitungszeit [ms]
Signalverarbeitung	FPGA	60
Hitverarbeitung	FPGA	35
Plotverarbeitung	FPGA	25
Trackverarbeitung	PowerPC	45
Gesamtverarbeitungszeit [ms]		**165**

Tabelle 4: Ergebnis des Validierungsverfahrens von VZ2 nach der Anforderungs- und System-Architektur-Änderung

Verarbeitungseinheit	durchschnittliche Prozessorauslastung [%]
FPGA	73
PowerPC	27

5. Vorteile und Erfahrungen

Die Vorteile des vorgestellten Ansatzes können wie folgt skizziert werden:

- Die Automatisierung des Ablaufs von Validierungsverfahren reduziert den Arbeits- und Zeitaufwand für jede Validierung.
- Der Architekt kann die Auswirkungen von Anforderungsänderungen (Impact-Analyse) durch deren Verbindung zu Validierungszielen bis zur System-Architektur nachverfolgen.
- Der Validierungsprozess zeigt dem Architekten nicht direkt ersichtliche Auswirkungen, die durch Änderungen an der System-Architektur entstehen.
- Das Abstraktionsniveau der Validierungsziele ist beim Entwurf und bei der Analyse von System-Architektur-Änderungen hilfreich, damit der Architekt nicht den Überblick durch die hohe Anzahl an Anforderungen und deren Verbindungen untereinander verliert.

- Die Vergleichswerte für das Validierungsverfahren eines Validierungsziels stehen in den zugeordneten Anforderungen. Das Validierungsziel und das dazugehörige Verfahren bleiben bei Änderungen an diesen unverändert (siehe Anforderungsänderung am Ende von Kapitel 4).
- Die standard-konforme Verwendung der UML erlaubt die Transformation und das Auslesen von Modellinformationen mit vorhandenen Technologien wie beispielsweise dem Eclipse Modeling Framework [EMF10], was den entstehenden Mehraufwand durch den Ansatz reduziert und die Nutzung des UML-Modells als Datenquelle für die Validierungsverfahren mit geringem Aufwand ermöglicht.
- Die Auftrennung in Entwicklungs- und Validierungsmodell erlaubt eine unabhängige Bearbeitung der Daten und verhindert die Vermischung von validierungs- und entwicklungsspezifischen Informationen.
- Der Validierungsprozess kann bereits zu Beginn oder vor dem Projektstart den Architekten bei der Abschätzung der Entwicklungszeit und -kosten unterstützen. Die Validierung verwendet die aktuell in der Entwicklung vorhandenen Anforderungen und Informationen und kann dem Fortschritt im Projekt angepasst werden.

Der Ansatz wurde anhand von Projekten der SOPHIST GmbH und mit Hilfe unserer Erfahrungen aus verschiedenen Entwicklungen eingebetteter und softwareintensiver Systeme entwickelt. In einem der Projekte werden mehr als 40 Verarbeitungsbausteine, Mikroprozessoren und Field Programmable Gate Arrays (FPGA), und über 60 Software-Komponenten eingesetzt. Die dafür eingesetzten Validierungsverfahren berücksichtigen deutlich mehr Einflussfaktoren (z.B. Datenmengen, Datenreduktion, Kommunikations-Overhead und -Latenzzeiten, Speicherzugriffe, Seiteneffekte durch kontinuierliche Signalverarbeitung) als in dem gezeigten Beispiel und auch das Validierungsmodell besteht nicht nur aus wenigen Elementen, die sich in einem Diagramm darstellen lassen. Unsere Erfahrungen zeigen, dass sich der durch den Ansatz verursachte Mehraufwand bereits nach wenigen Iterationen des Validierungsprozesses amortisiert.

6. Zusammenfassung und Ausblick

Die Entwicklung eingebetteter softwareintensiver Systeme ist mit sich ständig ändernden Bedingungen konfrontiert, was gerade den Architekten der System-Architektur, dem Fundament der eingebetteten Systeme, vor große Herausforderungen stellt. Zu seinen Aufgaben zählen unter anderem der Entwurf und die Dokumentation einer System-Architektur unter Berücksichtigung vieler Einflussfaktoren sowie die Validierung dieser gegenüber den Kundenanforderungen. Diese zeit- und arbeitsaufwendige Aufgabe sollte gerade bei der zunehmenden Komplexität der zu entwickelnden Systeme, hohe Anzahl an Anforderungen und starker Verbindung dieser untereinander, nicht auf der Methode des scharfen Hinsehens basieren.

Der in diesem Paper vorgestellte Ansatz beschreibt einen Validierungsprozess, der in das iterative Vorgehen beim Entwurf der System-Architektur eingebaut wird. Mit Hilfe von Validierungszielen werden architekturrelevante Anforderungen auf einem für den Entwurf der System-Architektur passenden Abstraktionsniveau beschrieben.

Die Validierung der System-Architektur wird aufgeteilt in die Validierung aller Validierungsziele. Jedes Validierungsziel wird durch ein vom Architekten festgelegtes Validierungsverfahren gegenüber den Anforderungen getestet. Die für die Validierungsverfahren benötigten Informationen werden auf Basis einer vom Architekten festzulegenden validierungszielspezifischen UML-Notation im Validierungsmodell, unabhängig von den entwicklungsspezifischen Daten im Entwicklungsmodell, dokumentiert. Die entstehende Dokumentation ermöglicht die Nachvollziehbarkeit von Änderungen an der System-Architektur. Der Ansatz reduziert den Zeit- und Arbeitsaufwand für die Validierung der System-Architektur, indem der Ablauf der Validierungsverfahren automatisiert wird. Als Datenquelle dient das Validierungsmodell, deren Informationen durch die standardkonforme Verwendung der UML mit geringem Aufwand ausgelesen werden können. Es lassen sich dadurch auch Daten zwischen Entwicklungs- und Validierungsmodell austauschen. Durch die Entkopplung der Validierungsziele von konkreten Werten in den Anforderungen, müssen diese wie bei Wertänderungen nicht angepasst werden. Der durch den Ansatz entstehende Mehraufwand amortisiert sich nach unseren Erfahrungen bereits nach wenigen Validierungen, durch die Unterstützung des Architekten beim Entwurf und bei der Analyse von Anforderungs- und Architekturänderungen.

Aktuell wird an der Implementierung eines grafischen Werkzeugs für die Verwaltung der Validierungsziele sowie deren Verbindungen zu den architekturrelevanten Anforderungen und den Validierungsverfahren gearbeitet. Die oft noch manuell ausgeführten Validierungsverfahren sollen automatisiert werden. Ziel ist die Verbesserung der Benutzerfreundlichkeit des Ansatzes. In der Zukunft werden wir uns auf Simulationen zur Validierung verschiedener, stark voneinander abhängiger Validierungsziele konzentrieren. Diese Simulationen sollen auf mehr oder weniger detaillierte Informationen in Abhängigkeit von der aktuellen Entwicklungsphase basieren und somit dem Fortschritt im Projekt angepasst werden können. Dabei werden auch die Möglichkeiten unterschiedlicher Datenquellen, innerhalb und außerhalb der UML sowie deren Einbindung in die modellgetriebene Validierung der System-Architektur untersucht.

Literaturverzeichnis

[Aqu11] PREEvision von aquintos, http://www.aquintos.com, letzter Zugriff 04.02.2011

[Bal02] Marc Balcer, Stephen Mellor, Executable UML: A foundation for Model-Driven Architecture, Addison-Wesley, 2002

[CMU11] Architecture Analysis & Design Language (AADL), Carnegie Mellon University, http://www.aadl.info, letzter Zugriff 03.01.2011

[CoF10] CoFluent Studio, http://www.cofluentdesign.com/, letzter Zugriff am 28.08.2010

[EMF10] Eclipse Modeling Framework (EMF), Eclipse Foundation, http://www.eclipse.org/emf/, letzter Zugriff am 28.08.2010

[Fei10] P. Feiler, L. Wrage, J. Hansson, System Architecture Virtual Integration: A Case Study, Embedded Real-time Software and Systems Conference, Mai 2010

[OMG10a] System Modeling Language (SysML), Object Management Group, http://www.omgsysml.org/, letzter Zugriff am 28.11.2010

[OMG10b] UML-Profil MARTE, http://www.omgwiki.org/marte/, Object Management Group, letzter Zugriff am 28.11.2010

[Spa10] Enterprise Architect, Sparx Systems, http://www.sparxsystems.com, letzter Zugriff am 28.11.2010

[UML10] Unified Modeling Language (UML) Version 2.3, http://www.uml.org, 2010