Fraunhofer
IESE

Systemeigenschaften dank Architektur fest im Griff

Autoren:
Martin Becker
Thorsten Keuler
Jens Knodel

IESE-Report Nr. 019.12/D
Version 1.0
Dezember 2012

Eine Publikation des Fraunhofer IESE

Das Fraunhofer IESE ist ein Institut der Fraunhofer-Gesellschaft.
Das Institut transferiert innovative Software-Entwicklungstechniken, -Methoden und -Werkzeuge in die industrielle Praxis. Es hilft Unternehmen, bedarfsgerechte Software-Kompetenzen aufzubauen und eine wettbewerbsfähige Marktposition zu erlangen.

Das Fraunhofer IESE steht unter der Leitung von
Prof. Dr. Dieter Rombach
(Geschäftsführender Institutsleiter)
Prof. Dr.-Ing. Peter Liggesmeyer
(Wissenschaftlicher Institutsleiter)
Fraunhofer-Platz 1
67663 Kaiserslautern

Abstract

Konkurrierende Eigenschaften von software-intensiven Systemen, wie funktionale Sicherheit, Performanz oder Wartbarkeit abzuwägen und miteinander zu koordinieren, zählen zu den Hauptherausforderungen bei der Entwicklung komplexer Systeme. Ein Architektur-zentriertes Vorgehen erlaubt es frühzeitig Lösungskonzepte auf ihre Tragfähigkeit zu untersuchen und über Entwicklungsphasen hinweg stetig auf konsistente Umsetzung zu überprüfen. In dem Beitrag wird eine Übersicht über typische Modelle und deren Einsatz bei der Entwicklung gegeben.

Inhaltsverzeichnis

1 Einleitung

Konkurrierende Eigenschaften von software-intensiven Eingebetteten Systemen, wie funktionale Sicherheit, Echtzeitfähigkeit oder Wartbarkeit abzuwägen und miteinander zu koordinieren, zählen zu den Hauptherausforderungen bei der Entwicklung komplexer Systeme. Gerade im Bereich der eingebetteten Systeme wird dem Software-Architekturentwurf und der daraus resultierenden Softwarearchitektur eine entscheidende Rolle bei der Bewältigung der oben genannten Herausforderungen beigemessen, da die Architektur die Brücke zwischen dem Problem- und dem Lösungsraum bildet und die Basis für alle nachfolgenden Aktivitäten darstellt.

Dies belegen auch Erfahrungen, die beim Fraunhofer IESE in mehr als 50 Software Architekturbewertungs- und -verbesserungsprojekten in verschiedenen Anwendungsbereichen gesammelt wurden. Eine Analyse dieser Projekte bestätigt zunächst, dass die Softwarearchitektur ein entscheidender Faktor ist, der die Erreichbarkeit von gewünschten Eigenschaften wie beispielsweise funktionale Sicherheit, Wartbarkeit, oder Wiederverwendbarkeit der Software und damit der Gesamtsysteme wesentlich beeinflusst.

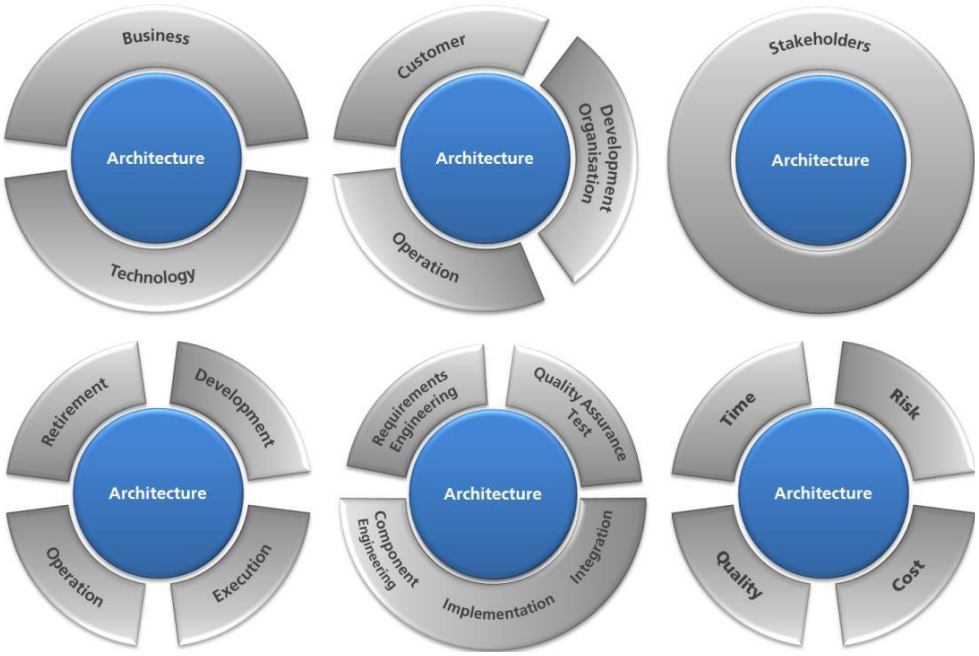
Ein tieferer Blick in die Projektpraxis offenbart jedoch, dass die wenigsten Unternehmen das volle Potenzial der Architektur nutzen, z.B. die Architekturbewertung als Instrument zur Qualitätssicherung. Oft wird auf historisch gewachsene Lösungen zurückgegriffen, die in der Vergangenheit gut funktioniert haben, jedoch ohne genau zu wissen, wie gut sie zu den aktuellen Anforderungen passen. Die Analysen zeigen dann oftmals, dass die Lösungskonzepte in der Implementierung nicht so umgesetzt sind, wie es eigentlich angedacht war. In letzter Konsequenz kann man sich nicht mehr darauf verlassen, dass die implementierte Software die Anforderungen erfüllen wird. In mindestens 80 % der betrachteten Projekte existierte keine aktuelle oder hinreichende Dokumentation der Architektur, um anstehende, kritische Entscheidungen treffen zu können. Bei zahlreichen Projekten mit mehreren Millionen Code-Zeilen wurde die Konformität zwischen Architektur (Soll-Architektur) überprüft. Unsere Werkzeuge konnten dabei in bestimmten Projekten mehr als 50.000 Architekturverletzungen entdecken.

Dieser Übersichtsartikel möchte das Bewusstsein für das oftmals ungenutzte Architekturpotential schärfen und Einstiegshilfen in die Architektur-zentrierte Entwicklung geben.

2 Architektur als zentraler Dreh- und Angelpunkt

Liegt die geplante Architektur eines Systems als explizites Modell vor, lassen sich darin konkurrierende Eigenschaften des Systems abwägen und miteinander koordinieren. Der Trend geht dabei, wie beispielsweise in [1] geschildert, in Richtung einer ganzheitlichen modellbasierten Entwicklung, wo die Modelle in maschinenverarbeitbarer Form vorliegen.

Je nach Modellumfang lassen sich, wie in der folgenden Illustration dargestellt, unterschiedliche Sachverhalte und Interessen über die Architektur miteinander in Beziehung setzen und in Einklang bringen.



Aufgrund der Kritikalität der Architektur ist es insbesondere wichtig, die Auswirkungen von architekturrelevanten Design-Entscheidungen zu kennen, um Vor- und Nachteile bewerten zu können. So bietet das Architektur-Modell die Grundlage, frühzeitig Lösungskonzepte auf ihre Tragfähigkeit zu untersuchen und über Entwicklungsphasen hinweg stetig auf konsistente Umsetzung zu überprüfen. Die Tragfähigkeitsuntersuchungen führen insbesondere auch dazu, dass Fehlentscheidungen früh erkannt werden und nicht erst später kostspielig revidiert werden müssen. Aufgrund des omnipräsenten Risikos mögli-

cher Fehlentscheidungen ist es unerlässlich, die Entwurfsentscheidungen im Rahmen von Architekturbewertungen in ihrer Gesamtheit zu betrachten und zu beurteilen, um mögliche Probleme und Risiken zu identifizieren.

Im Kontext varianten-reicher Systeme, wie z.B. Plattformen oder Produktlinien, ist die Architektur das zentrale Modell, um einen Überblick über Gemeinsamkeiten und Unterschiede der Varianten zu geben und die effiziente Produktion der Varianten zu gewährleisten. Sie ermöglicht die Wiederverwendung-im-Großen.

Im Paradigma der Architektur-zentrierten Entwicklung [2] ist die Software-Architektur keine Phase, sondern eine kontinuierliche Aktivität, die innerhalb des Lebenszyklus der Software mit unterschiedlichen Zielstellungen ausgeführt wird. Die Architektur hat Schnittstellen zu den typischen Entwicklungsaktivitäten im Lebenszyklus der Software, wie Anforderungsmanagement, Komponententenentwurf, Implementierung, sowie dem Testen und der Integration der Software.

3 Typische Architektur-Sichten Eigebetteter Systeme

Aufgrund der inhärenten Komplexität von Architektur-Modellen, zählt es zur gängigen Praxis, dass Architekturen über eine Menge vorab definierter Sichten beschrieben werden, die sich an den Bedürfnissen der einzelnen Stakeholder und den Nutzungsszenarien der Architekturmodelle ausrichten. Nur so kann gewährleistet werden, dass die erforderlichen Informationen in einer geeigneten Präsentation vorliegen und nicht benötigte Information keine unnötigen Dokumentations- und Wartungsaufwände verursacht.

Auch wenn man im Allgemeinen die benutzten Architektursichten auf die adressierten Anforderungen, die Systemart, sowie die Stakeholder im Detail ausrichten muss, bilden die folgenden Sichten eine gute Ausgangsbasis für den Einstieg in die Architektur-zentrierte Entwicklung: Kontextsicht, Bausteinsicht, Dynamische Sicht, sowie Verteilungssicht, die nachfolgend näher beschrieben werden.

3.1 Kontextsicht

Die Kontextsicht dient zur Beschreibung eines Systems als „Blackbox" in seinem Ausführungskontext. Das heißt, die Kontextsicht zeigt das eingebettete System wie es mit anderen Akteuren zusammenhängt. Akteure können dabei externe Systeme sein, Sensoren, Aktuatoren, oder aber auch Benutzer. Im Falle von eingebetteten Systemen liegt zudem ein klarer Fokus auf Datenflüssen, die zwischen Systemen und Systembestandteilen ausgetauscht und bearbeitet werden.

Die Daten-zentrische Kontextsicht eingebetteter Software Architekturen verfolgt damit die folgenden Ziele:

- Übersicht über das eingebettete Systems und seine Umgebung
- Abgrenzung des Systems gegenüber externen Akteuren
- Explizite Darstellung der Abhängigkeiten des Systems von bestimmten Kontextfaktoren

Im Beispiel sieht man den Systemkontext eines Systems „ACC", das mit zwei externen Systemen verbunden ist.

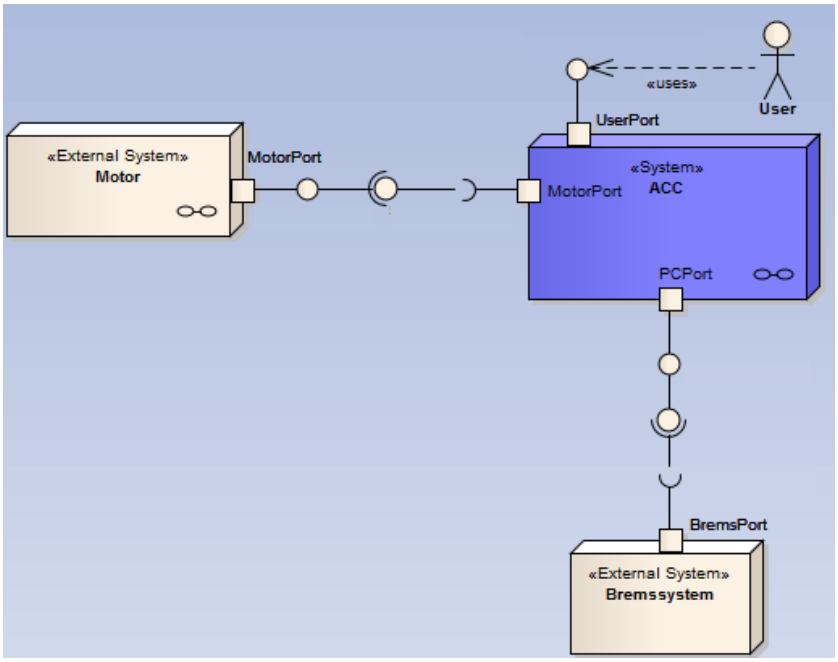


Abbildung 1 Beispiel für eine Kontextsicht

3.2 Bausteinsicht

Im Gegensatz zur Kontextsicht, die das System als „Blackbox" darstellt, zeigt die Bausteinsicht die interne Organisation des Systems mit funktionalen Einheiten, die sogenannten Bausteine. In eingebetteten Systemen arbeiten diese Bausteine häufig auf einem kontinuierlichen Datenstrom und liefern als Ausgabe wiederum Daten, die von anderen Bausteinen genutzt und weiterverarbeitet werden. Die Bausteinsicht kann über Anreicherung mit dynamischer Information als Grundlage zur Analyse von Laufzeiteigenschaften herangezogen werden. Dazu werden die Bausteine selbst oder aber auch die Verbindungen zwischen den Bausteinen mit Hilfe von Verhaltensdiagrammen modelliert.

Im Kontext der Modellierung eingebetteter Software Architekturen verfolgt die Bausteinsicht die folgenden Ziele:

- Übersicht über den internen funktionalen Aufbau des Systems
- Funktionale Hierarchisierung des Systems zur Beherrschung der Komplexität
- Bewertung und Vorhersage von Laufzeiteigenschaften

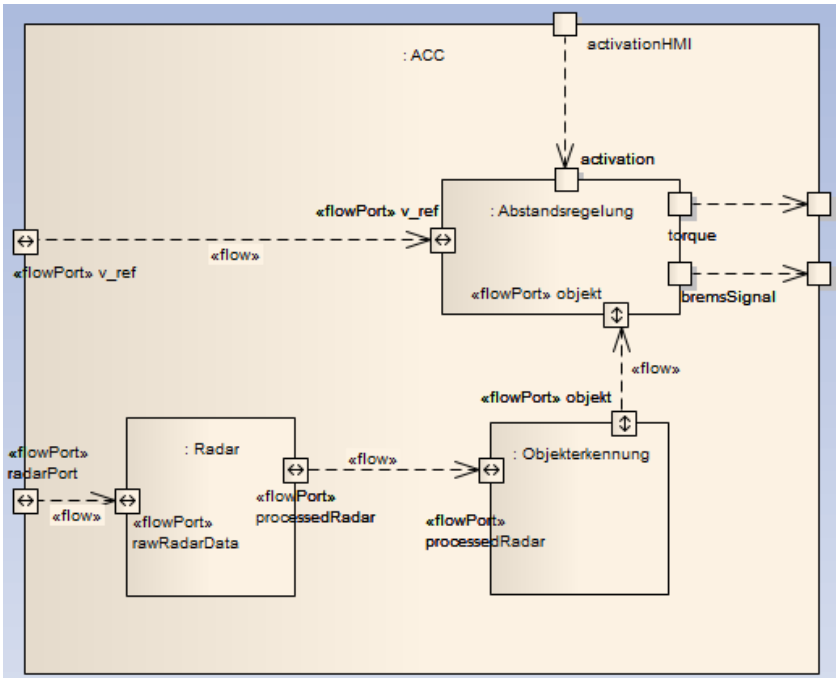


Abbildung 2 Bausteinsicht

3.3 Dynamische Sicht

Wenn Verhalten bezüglich der Kommunikation zwischen Blöcken oder auch Verhalten der Blöcke selbst modelliert werden soll, werden Zustandsdiagramme, Sequenzdiagramme, sowie Aktivitätsdiagramme eingesetzt. Welches der Diagramme zum Einsatz kommt hängt letztendlich von der Zielstellung des Modellteils ab. Sollen sichtbare Zustände gezeigt werden, dann sind Zustandsdiagramme am sinnvollsten. Wenn algorithmische Eigenschaften modelliert werden sollen, dann werden Aktivitätsdiagramme eingesetzt. Sequenzdiagramme sind von Vorteil, wenn die Darstellung der konkreten Abfolge beim Informationsaustausch relevant ist.

In der folgenden Abbildung ist ein beispielhaftes Zustandsdiagramm dargestellt, welches die extern sichtbaren Zustände des ACC Systems beschreibt.

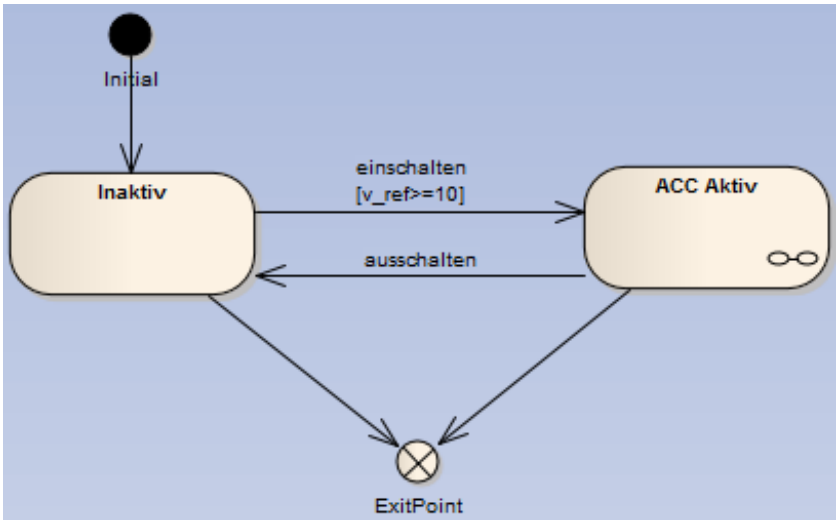


Abbildung 3 Zustandsdiagramm zur Modellierung von Verhalten

3.4 Verteilungssicht

Die Verteilungssicht zeigt die Relation der Softwareanteile bezüglich der Hardwareumgebung und deren Topologie. Im Kontext der Modellierung eingebetteter Software Architekturen verfolgt die Verteilungssicht daher die folgenden Ziele:

- Übersicht über die Hardware Topologie
- Zuweisung von Software zu Hardwareeinheiten
- Übersicht der Hardwareeigenschaften, wie Prozessor oder Speicher

In der folgenden Abbildung ist beispielhaft eine Verteilungssicht dargestellt:

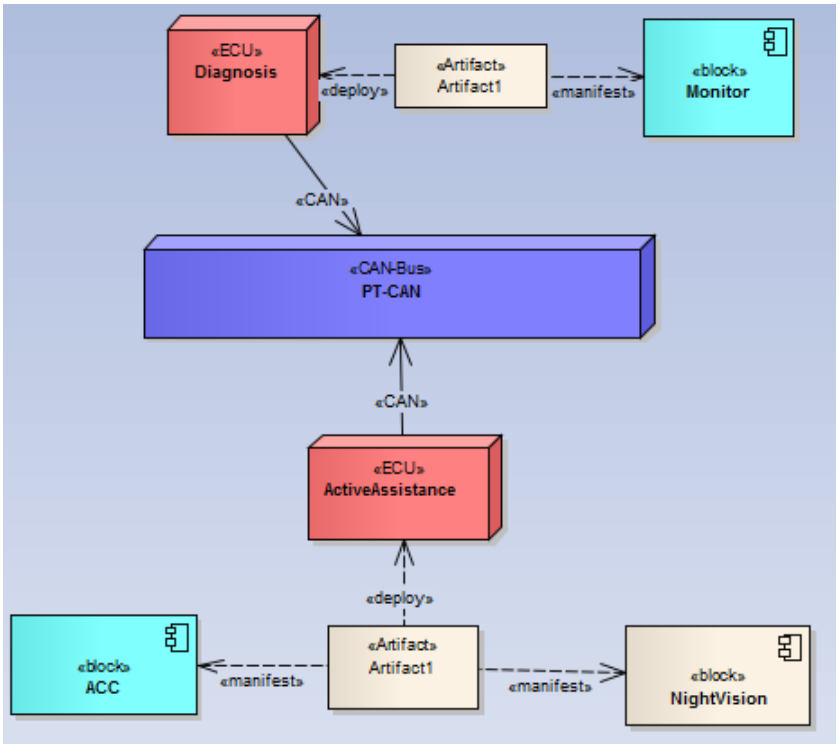


Abbildung 4 Verteilungssicht eines eingebetteten Systems

4 Wege in die Zukunft

Architekturen ebnen den Weg zu qualitativ hochwertigen Software. Insbesondere in der Zusammenarbeit zwischen OEMs und Zuliefern wird die Architektur immer mehr Bedeutung beigemessen werden, da sie die Grundlage für die Beherrschbarkeit des Gesamtsystems bildet. Außerdem kann die Architektur auch für vertragliche Absprachen herangezogen werden. Unsere Erfahrungen zeigen, dass insbesondere die systematische Bewertung der Architektur einen wirklichen Mehrwert für die Softwareentwicklung bringt. Sie deckt potenzielle Probleme frühzeitig auf und zahlt sich unserer Erfahrung nach fast immer aus.

Literatur

[1] M. Broy: Mit welcher Software fährt das Auto der Zukunft?, ATZ extra "125 Jahre Automobil", April 2011, S. 92-97

[2] T. Keuler, J. Knodel, M. Naab "Architecture-Centric Software and Systems Engineering", Technical Report, Fraunhofer IESE, 2011

Dokumenten Information

Titel:	Systemeigenschaften dank Architektur fest im Griff
Datum:	Dezember 2012
Report:	IESE-019.12/D
Status:	Final
Klassifikation:	Public Unlimited