Ist Architektur-Entwicklung überhaupt noch zeitgemäß?
Und wenn ja, wie machen wir das heute richtig?

Jens Liebehenschel, Frankfurt University of Applied Sciences

Am Anfang dieser Arbeit erfolgt eine kurze Begriffserklärung. Der Nutzen von frühzeitiger Architektur-Entwicklung wird dargestellt. Nach der Vorstellung eines praxiserprobten und in sich schlüssigen Basissatzes von (Werkzeug-unabhängigen) Architektursichten für System- und Software-Architekturen für eingebettete Systeme endet die Arbeit mit Tipps zur Architektur-Entwicklung.

Begriffliches

Architektur eines Software-Systems beschreibt die essentiellen Aspekte. Dazu gehören neben Struktur und Verhalten insbesondere wichtige Entscheidungen, zum Beispiel verwendete Konzepte und Technologien.
Jedes System besitzt eine Architektur. Manchmal ist sie gezielt entwickelt worden, manchmal einfach so entstanden – oft eine Kombination von beidem.
Die Architektur ist notwendig zur Erfüllung qualitativer Anforderungen, also zum Beispiel Performance, Speicherplatzbedarf und Wartbarkeit, aber auch Sicherheit, sowohl Safety als auch Security. Basierend auf den Anforderungen müssen Konzepte gefunden werden, die den gewünschten Trade-Off zwischen allen qualitativen Anforderungen möglichst gut erfüllen. Die Erfüllbarkeit wird auf Architekturebene erreicht, nicht in einer „Performance-“ oder „Safety-Komponente“.

Die Architektur-Dokumentation dient neben der Entwicklung auch der Kommunikation. Sie kann vielfältig ausgestaltet sein. Heute üblich ist der Einsatz von Tools, die nach dem Modell-Sichten-Paradigma arbeiten. Jedes Element im System existiert nur genau einmal, dies verhindert Redundanz.
Die Dokumentation beschreibt die Architektur für alle Stakeholder in einer sinnvollen Weise. Wenn eine Architektur nicht dokumentiert ist, ist zumindest ihre Sichtbarkeit vermindert. Am Code ist die Architektur definitiv nicht zu erkennen.

Warum wird Architektur-Entwicklung vernachlässigt?

Eine schlechte Architektur(-Dokumentation) hat einen geringen Nutzen. Wenn der Nutzen im Verhältnis zum Aufwand nicht hoch genug ist, dann erhält das Thema keine hohe Priorität.
Metriken beeinflussen Verhalten. Zu bestimmten Meilensteinen müssen (wegen Standards, Prozessen, etc.) Architektur-Arbeitsprodukte vorhanden sein. Diese werden mit möglichst geringem Aufwand erstellt. Die Metrik wird erfüllt, leider oft mit geringem Nutzen.

Wobei und wie unterstützt Architektur-Entwicklung?

Anders als Gegenständliches können wir Software-Systeme mit unseren Sinnen nicht erfassen. Ein sinnvoller Zugang in Form einer geeigneten Darstellung wird benötigt. State-of-the-art ist dies ein Architekturmodell. Wenn die Architektur entwickelt wird, braucht das Team eine Diskussionsgrundlage. Ein gutes Modell gibt einen verständlichen Überblick über das zu entwickelnde System und hilft dabei, die richtigen Abstraktionen zu finden und zu dokumentieren, also beispielsweise Schnittstellen und Verantwortlichkeiten.

Inhalt der Architektur-Dokumentation
Das Architekturmodell enthält Struktur-, Verhaltens- und weitere Sichten. Die Stakeholder entscheiden über die Sichten und dort sichtbaren Aspekte. Somit hängt die individuelle Ausprägung vom Projekt ab.
Dennoch hat der Autor in verschiedenen Projekten festgestellt, dass ein sinnvoller Satz von Sichten immer wieder – wenn auch angepasst – zum Einsatz kommt. Die notwendige Anpassung betrifft neben der Tiefe und Detaillierung der Modellierung auch die vorhandenen Domänen wie Software, Hardware und Hydraulik.

Architektursichten für eingebettete Systeme
Im Folgenden werden zunächst Strukturansichten (hierarchische Dekomposition), dann Verhaltenssichten und weitere Sichten gezeigt. Die Modellierung erfolgt signalfluss-orientiert. Daher wird sinnvollerweise die SysML eingesetzt.
Das Beispielsystem ist bewusst sehr einfach gewählt, so dass die Diagramme gut verständlich sind, eine Übertragbarkeit aber möglich ist.

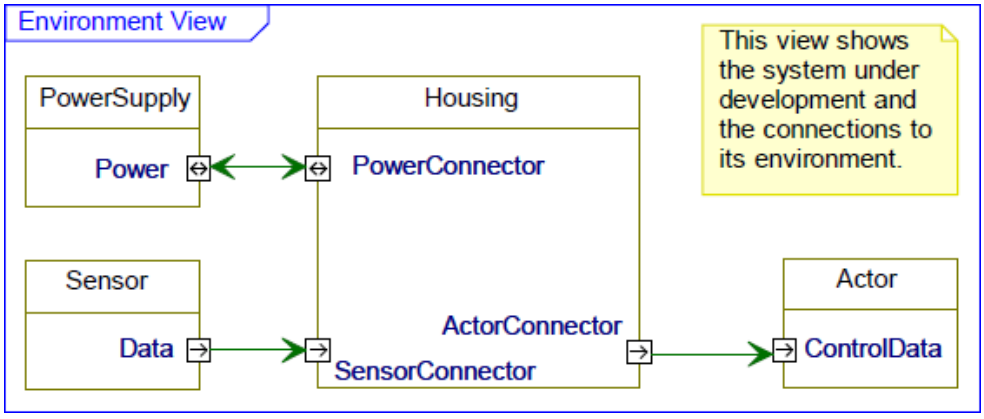

Abb. 1: Systemumgebung

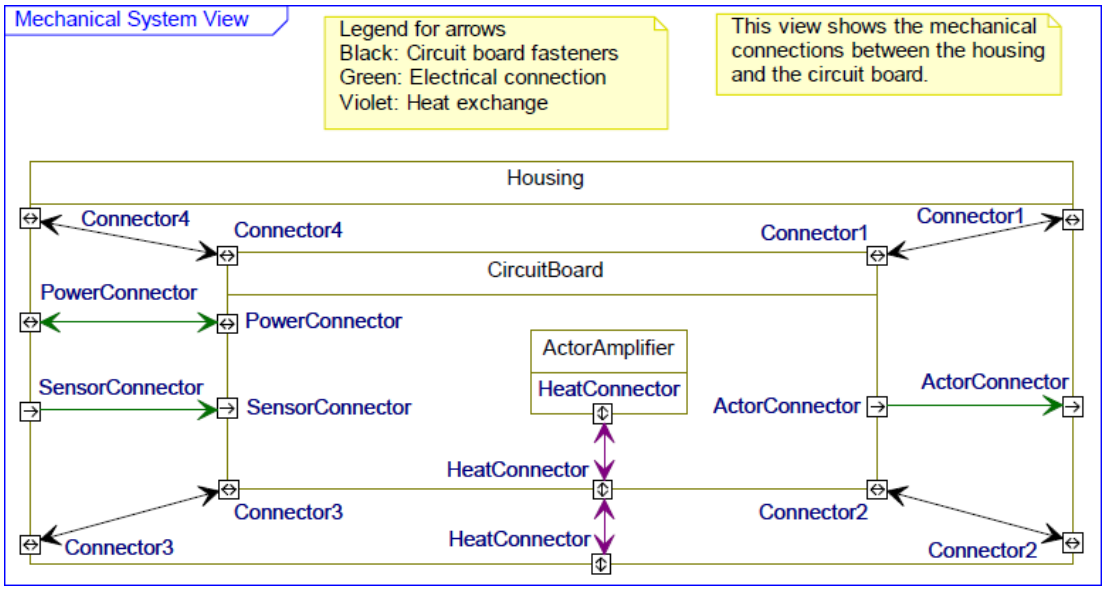

Abb. 2: Gehäuse und Platine mit Wärmequelle und Befestigungen

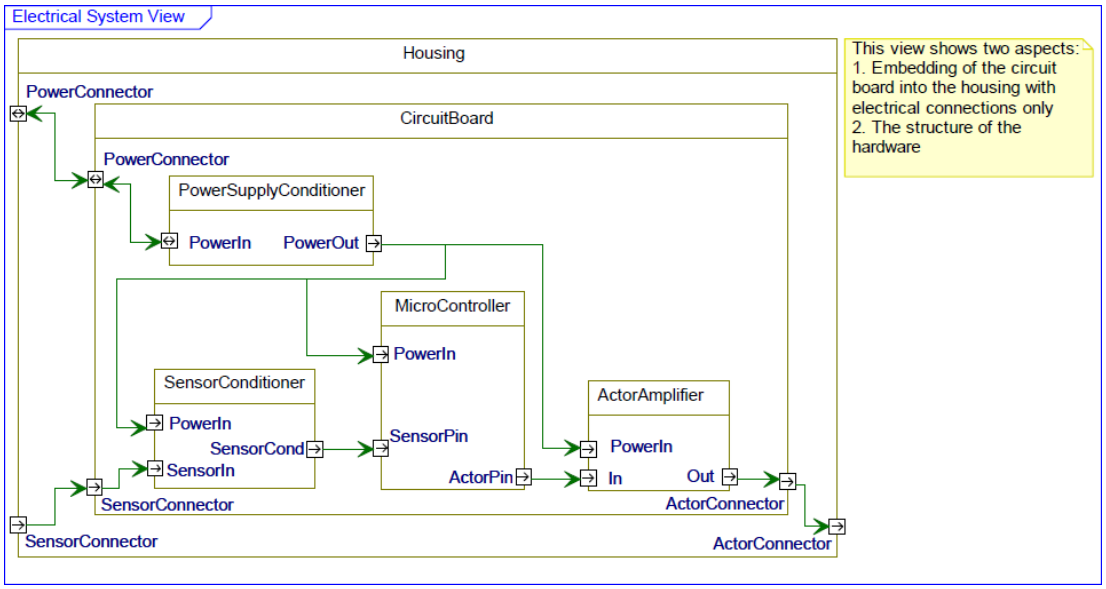

Abb. 3: Elektrisches System

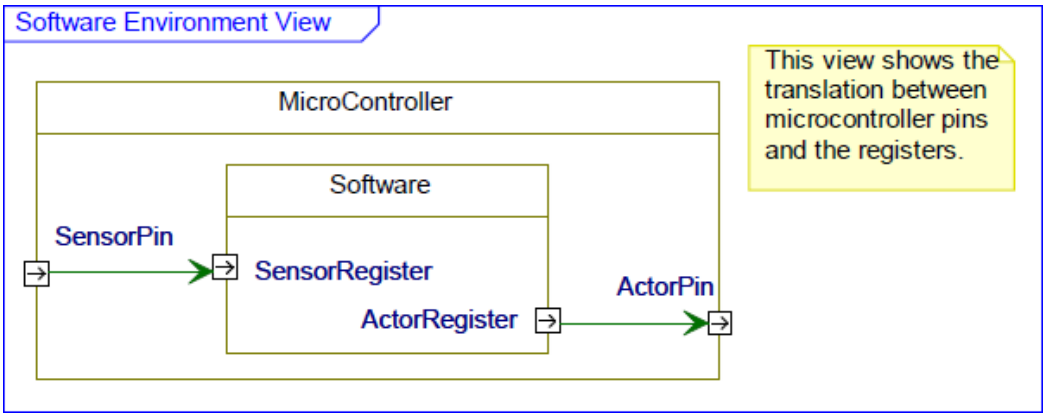

Abb. 4: Mikrocontroller als Ausführungsumgebung für die Software

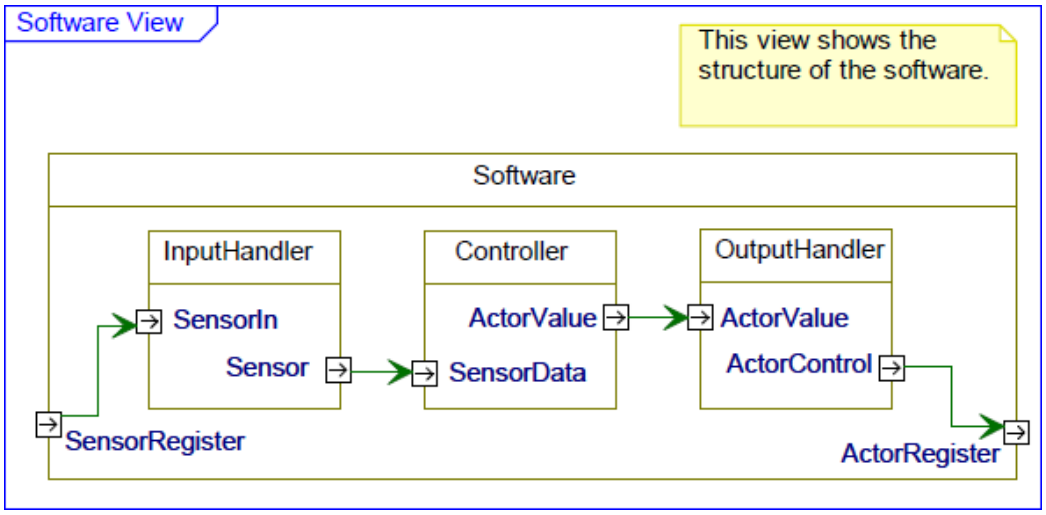

Abb. 5: Software und Top-Level Software-Komponenten

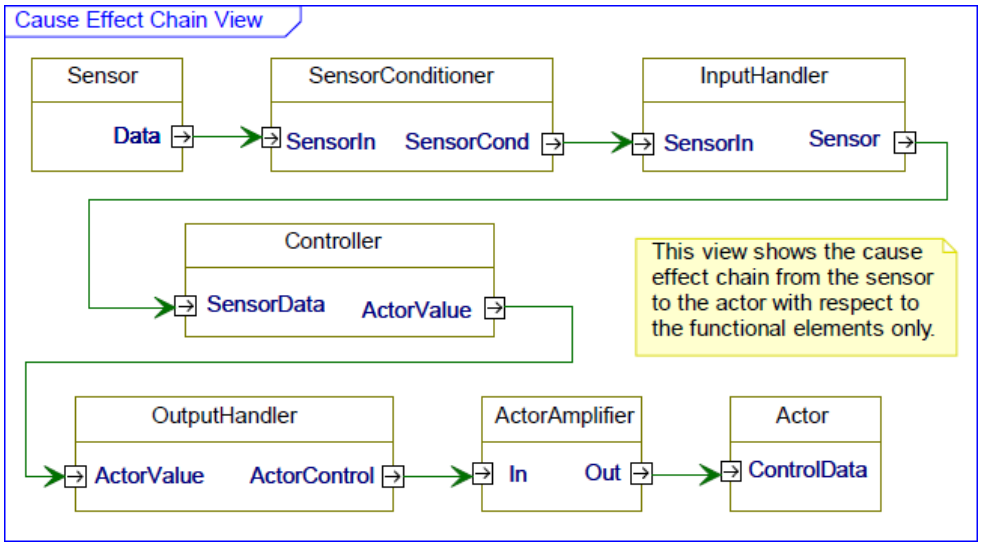

Abb. 6: Eine Wirkkette über Domänengrenzen

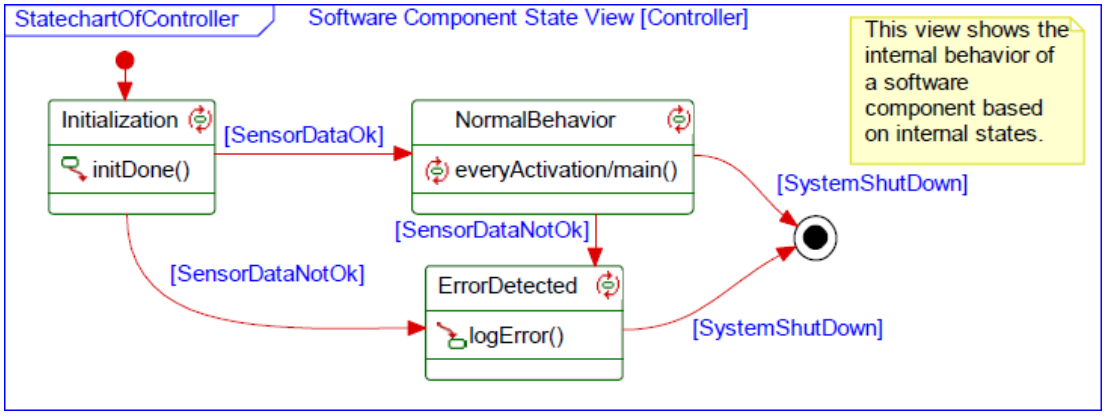

Abb. 7: Interne Zustände einer Software-Komponente

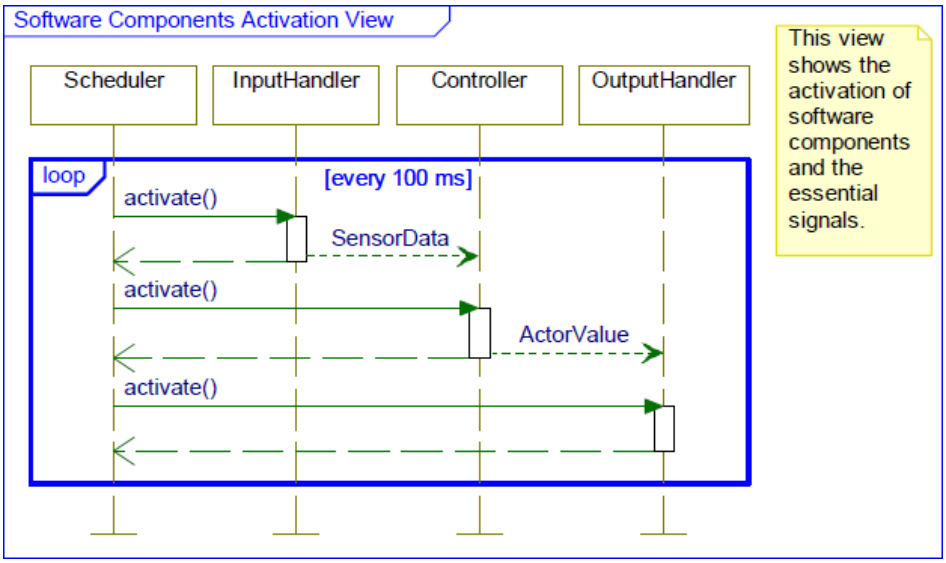

Abb. 8: Aktivierung der Software-Komponenten mit essentiellen Datenflüssen

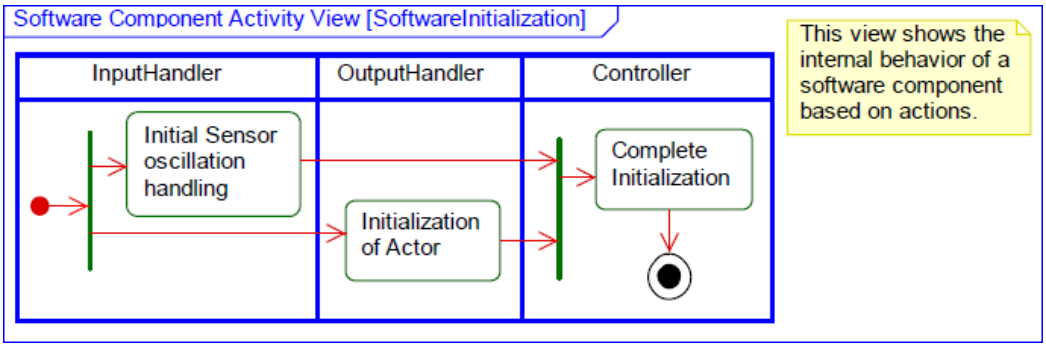

Abb. 9: Aktivitätsdiagramm zur Modellierung von Nebenläufigkeit

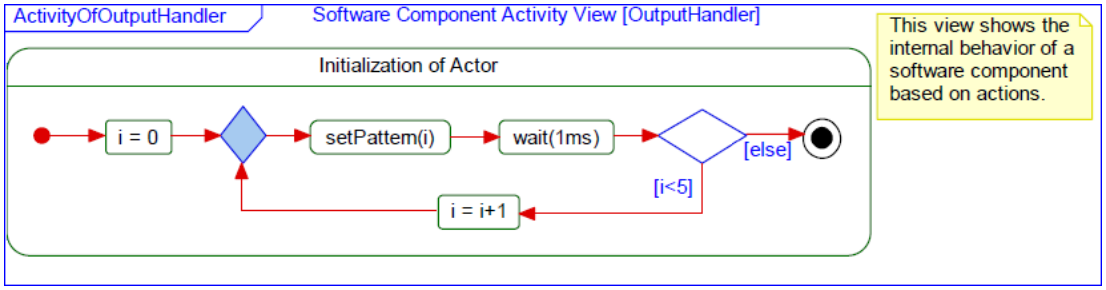

Abb. 10: Aktivitätsdiagramm zur Modellierung eines Algorithmus

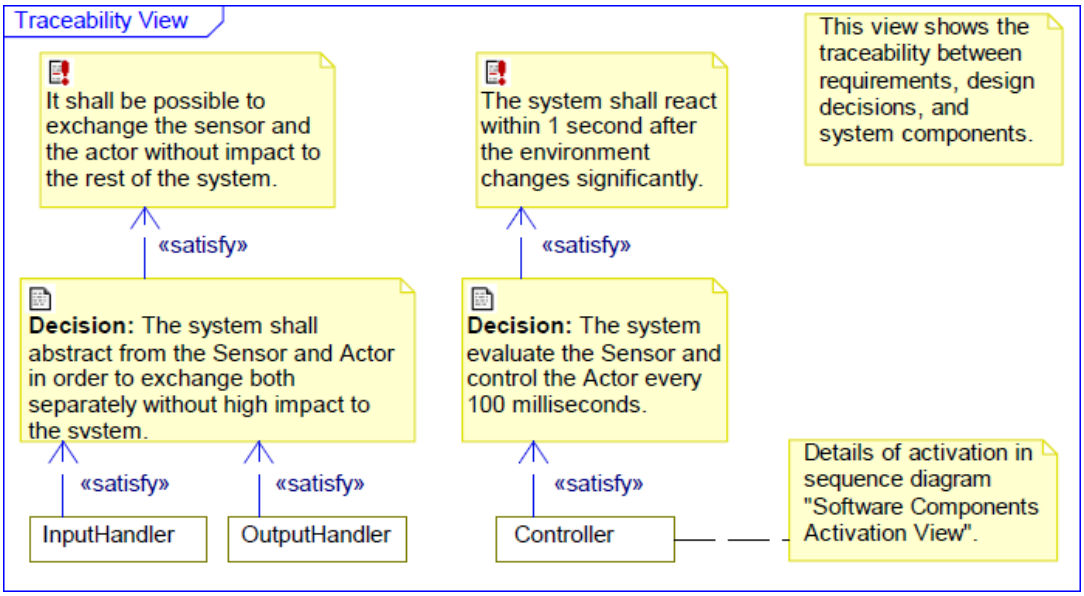

Abb. 11: Traceability mit Anforderungen und Design-Entscheidungen

Je nach Interessen der Stakeholder können weitere Sichten sinnvoll sein, wie Verantwortlichkeiten der Komponenten oder Testfälle.
Für weitere wichtige Aspekte der Entwicklung kann das Architekturmodell Anknüpfungspunkte bieten und Entscheidungsgrundlage sein, zum Beispiel für Laufzeitbetrachtungen, Schedulability-Analysen und Multicore-Betrachtungen.

Modellierungstipps und Erfahrungsberichte

Dieser Abschnitt enthält einige Tipps zur Erstellung von Architekturmodellen. Selbstverständlich müssen diese an die Gegebenheiten des Systems und des Entwicklungsprojekts angepasst werden.

- Eine oder mehrere Einstiegsseiten ins Modell für unterschiedliche Interessen der verschiedenen Stakeholder helfen beim Zurechtfinden im Modell und erhöhen die Akzeptanz.
- Modellierungssprachen wie die SysML besitzen einen hohen Sprachumfang. Oft wird nur eine kleine Teilmenge benötigt, um sinnvolle Modelle zu bauen. Je weniger Features der Sprachen verwendet werden, desto einfacher sind die Modelle zu verstehen.
- Eine Legende mit Erklärung der wesentlichen Modellierungselemente unterstützt das Verständnis des Modells bei Stakeholdern, die nicht über dieses Wissen verfügen.
- Jedes Diagramm sollte eine Information enthalten, aus der in wenigen Worten sichtbare Aspekte, Nutzen und/oder Zielgruppe hervorgehen.
- Die Anzahl der Sichten sollte so gering wie möglich sein, aber natürlich so groß wie nötig. Bei ihrer Auswahl muss auch der Pflegeaufwand (Versionen und Varianten) berücksichtigt werden.
- Bei der Tiefe der Modellierung (zum Beispiel bei Signalflüssen) muss ebenfalls der Pflegeaufwand berücksichtigt werden.
- Die bekannte Zahl „7+/–2“ (Anzahl der Informationseinheiten, die wir im Kurzzeitgedächtnis halten können) ist für die Anzahl der Elemente in Sichten nicht unbedingt relevant. Übersichtsdiagramme mit allen Komponenten sind oft auch dann sinnvoll und nützlich, wenn sie deutlich mehr Komponenten enthalten.
- In einer Sicht sollte üblicherweise nur eine Ebene der hierarchischen Dekomposition enthalten sein.
- Es tauchen immer wieder spezielle Aspekte auf, deren Modellierung kompliziert ist. Ein Modell muss nicht immer alles enthalten. Manchmal ist eine Notiz besser als aufwendige Modellierung.
- In einem Diagramm sollten generell nicht zu viele Aspekte vermischt werden. Besser Aspekte trennen, wenn dies sinnvoll möglich ist. Beispielsweise können in längeren Sequenzdiagrammen immer wiederkehrende Teile in separate Diagramme ausgelagert werden.
- Keinesfalls darf Struktur und Verhalten in einem Diagramm vermischt werden, auch wenn manche Modellierungstools dies zulassen.
- Jedes Element darf nur einmal im Modell enthalten sein. Sollte auf diesem Weg Redundanz eingeführt werden, dann funktioniert das Modell-Sichten-Paradigma nicht mehr.
- Die „Architektur-Bilder“ an der Wand sollten für alle Stakeholder nützlich sein, insbesondere der Entwicklung als Diskussionsbasis dienen.
- Die Dokumentation von essentiellen Entscheidungen ist wichtig. Oft können Alternativen qualitativ evaluiert und dennoch eine belastbare Entscheidung getroffen werden. Dies gilt auch für Qualitäten, die im weiteren Verlauf der Entwicklung quantifiziert werden müssen, wie beispielsweise die Safety.
- Auch wichtige methodische Entscheidungen sollten wegen der Nachvollziehbarkeit dokumentiert werden, zum Beispiel welche Sichten aus welchen Gründen existieren und welche Aspekte für welche Stakeholder wichtig sind.

- Wenn möglich sollte das Team an einem Modell arbeiten. Die parallele Bearbeitbarkeit kann üblicherweise erreicht werden. Der Nutzen eines Modells liegt hauptsächlich in der Konsistenz und Durchgängigkeit.
- Ganz wichtig ist in der Entwicklung generell die Konsistenz der Dokumentation. Innerhalb eines Modells sorgt ein gutes Werkzeug schon für Konsistenz, sofern es richtig eingesetzt wird. Die Konsistenz des Modells zu anderen Arbeitsprodukten kann meist über individuelle Programme besser als durch Reviews sichergestellt werden.
- Auch bei der Verbindung von Werkzeugen muss Automatisierung (eingebaute oder individuell entwickelte) eingesetzt werden.
- Bei der Entwicklung von Architekturmodellen gibt es Vorgaben. Wer Vorgaben definiert, sollte auch in die Arbeit involviert sein. Nur so kann man das Funktionieren der methodischen Ansätze sicherstellen. Kurze Feedback-schleifen sind in der Methodenentwicklung notwendig.

Zusammenfassung

Architektur-Entwicklung ist ein essentieller Bestandteil der System- und Software-Entwicklung. Je später das Projekt damit beginnt, desto geringer ist der Nutzen. Eine Nachdokumentation des entstandenen Systems hilft der Entwicklung nicht mehr. Die Architektur-Entwicklung und -Dokumentation sollte direkt am Anfang des Projekts begonnen und kontinuierlich während der Projektlaufzeit weiter betrieben werden. Sie muss als Basis der Entscheidungen dienen.
Damit die relevanten Stakeholder einen Nutzen der Architektur-Dokumentation haben, müssen diese gefunden und befragt werden. Nach den Interviews muss abgewogen werden, welche Aspekte zu welchem Zeitpunkt dokumentiert werden. Das Modell muss die Stakeholder unterstützen.

Autor

Prof. Dr. Jens Liebehenschel verfügt über langjährige Erfahrung in der Methoden- und Produktentwicklung variantenreicher Software-Systeme.
Er arbeitet seit 2002 an vielen verschiedenen Produkten im Automotive-Bereich.
Seit 2015 ist er an der Frankfurt University of Applied Sciences Professor für Mobile Systems Engineering.

Kontakt

E-Mail: liebehenschel@fb2.fra-uas.de