Software Design Needs Software Architecture

Liefert die UML uns dazu die richtigen Mittel?

Jürgen Hartung, Kölsch & Altmann Software & Management Consulting GmbH

Die Verbindung von Design und Architektur zwecks Nachverfolgbarkeit hinsichtlich Anforderungen und deren Umsetzung(en) wird durch Aspekte wie Safety (ISO 26262) oder bestimmten Entwicklungsmethoden (Automotive SPICE) forciert, wie sie z.B. in der Automobilindustrie gelten.
Werkzeuge auf Basis von UML, wie zum Beispiel Enterprise Architect, bieten einen einfachen Weg, diese Verbindungen abzubilden.
Die UML als Modellierungssprache besitzt jedoch keine Standardelemente oder gar eine Methode, wie die in der Software häufig vorkommende Varianten-bildung bzw. deren Produktlinien dargestellt werden können. Auch die während der Entwicklung entstehenden Versionen und ihre integre Komposition sind nicht Bestandteil von UML.

Einleitung

Die Unified Modeling Language (UML) mit ihren Sprachelementen wurde nicht auf die Belange einer funktionsorientierten Softwareentwicklung ausgelegt. Programmiersprachen wie C sind aber immer noch häufig im Einsatz.
Viele Projekte, speziell aus dem Embedded Umfeld, müssen aber immer mehr Wert auf vollständige Dokumentation und Nachvollziehbarkeit legen. Die Dokumentation der Architektur und der Entscheidungen, welche zu der gewählten Architektur führten, sind zu erstellen. Das daraus abgeleitete Design der Software darf dann wieder mit der Architektur verknüpft werden, um die Herkunft einzelner Software-komponenten zu belegen und den Zusammenhang darzustellen.

Das kommerzielle Produkt Enterprise Architect der Firma Sparx Systems [1] ist ein vollwertiges Werkzeug für die Nutzung der UML. Das Reverse Engineering im Werkzeug bietet Projekten die schnelle Erfassung der statischen Struktur in einem Modell. Einmal erfasste Modelle können dann im Werkzeug weiter entwickelt werden. Per Code-Generierung sind Änderungen im Modell dann wieder im Code zu finden.

UML für die Softwareentwicklung

Mit der UML wurden erstmals einheitliche Standards für die Beschreibung von Software eingeführt. Hervorzuheben ist hier die grafische Darstellung von statischen und vor allem dynamischen Aspekten. Zustandsdiagramm, Sequenzdiagramm und Aktivitätsdiagramm sind beliebte Diagrammtypen.
Die UML ist aber keine Methodik zur Softwareentwicklung. Eine Einführung der UML ersetzt nicht eine Entwicklungsmethode, wie z.B. das V-Modell [2] oder den Unified Process [3], bietet jedoch hilfreiche Unterstützung beim Entwicklungsprozess in den einzelnen Phasen.

Anforderungen an die Softwareentwicklung
Aus all den diversen Anforderungen, die es an die Softwareentwicklung gibt, werden nun die nachfolgenden Aspekte näher betrachtet:

1. Architektur
2. Design
3. Varianten

Architektur
Eine Softwareentwicklung benötigt am Anfang eine Architektur des Systems und der dafür zu entwickelnden Software. Neben statischen und dynamischen Aspekten sind hier noch diverse weitere Entscheidungsgrundlagen für oder gegen eine bestimmte Architektur zu dokumentieren.
Ein gelungenes Beispiel und Modell für die Dokumentation der diversen Aspekte einer Architektur findet sich unter dem Namen arc42 [4].

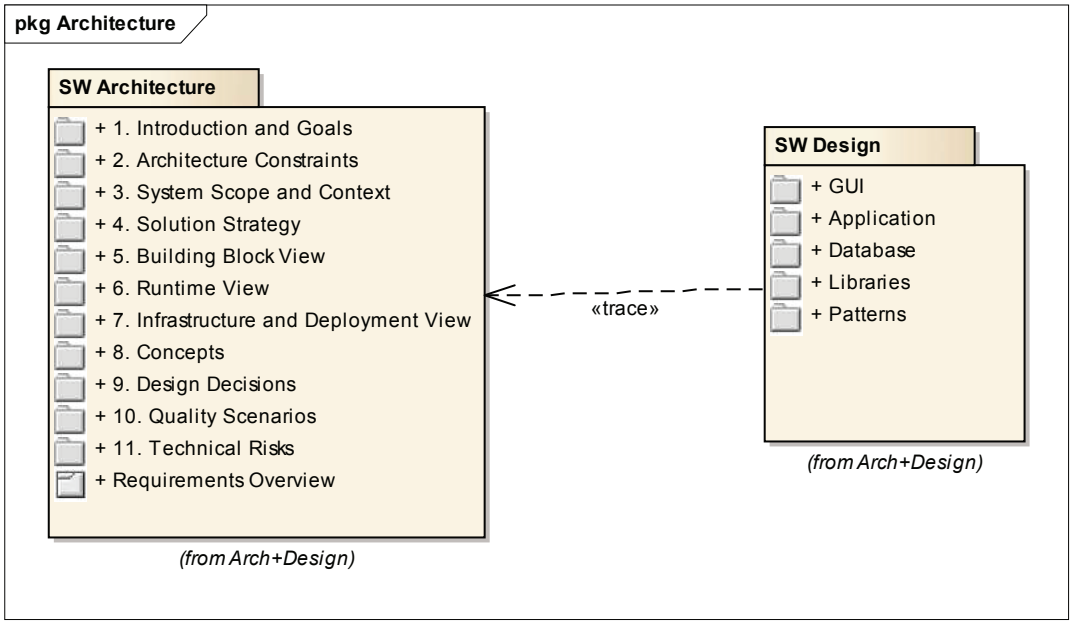

Abb. 1: Beispiel zur Dokumentation einer Architektur

Design
Im Design sind im Gegensatz zur Architektur viele Softwareteile in allen Details beschrieben. Auch die Partitionierung der Software muss damit genauer beschrieben sein, als es in der Architektur der Fall sein muss.
Im Vordergrund steht die Generierung von Code. Nur damit kann verhindert werden, dass Implementierung und die Beschreibung im Lauf der Softwareentwicklung auseinanderdriften.

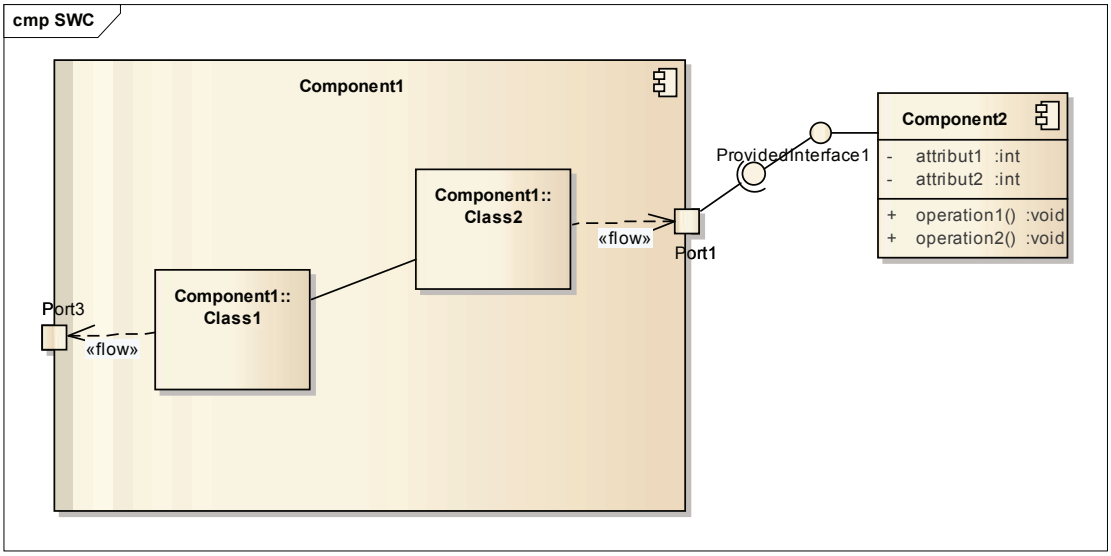

Abb. 2: Beispiel zur Dokumentation eines Design

Varianten
Gibt es von einem Produkt, wie zum Beispiel einem Steuergerät, verschiedene Ausprägungen, spricht man von Varianten. In der Softwareentwicklung bedeutet dies, dass es für ein oder mehrere Teile der Software unterschiedliche Implementierungen gibt.

Bei Verwendung der UML muss diese Abbildung im Modell geschehen. Entweder wird in einem Modellteil die gesamte Funktionalität über alle Varianten dargestellt oder in dem Modell ist immer nur die für die Generierung gültige Variante geladen. Es gibt einige Ansätze [5][6], wie die UML über eigene Sprachmittel die Variantenbildung unterstützt. Die verschiedenen Ansätze zeigen aber schon, dass es von Seiten der UML keine vordefinierte Handlungsweise oder vordefinierte Sprachelemente gibt.

Enterprise Architect als Werkzeug für die Verwendung der UML
Das Werkzeug Enterprise Architect verwaltet Modellelemente in einer Explorer-ähnlichen Struktur. Zur weiteren Separierung können in einem Gesamtmodell, typischerweise gleichbedeutend der Projektdatei, diverse Modelle enthalten sein.

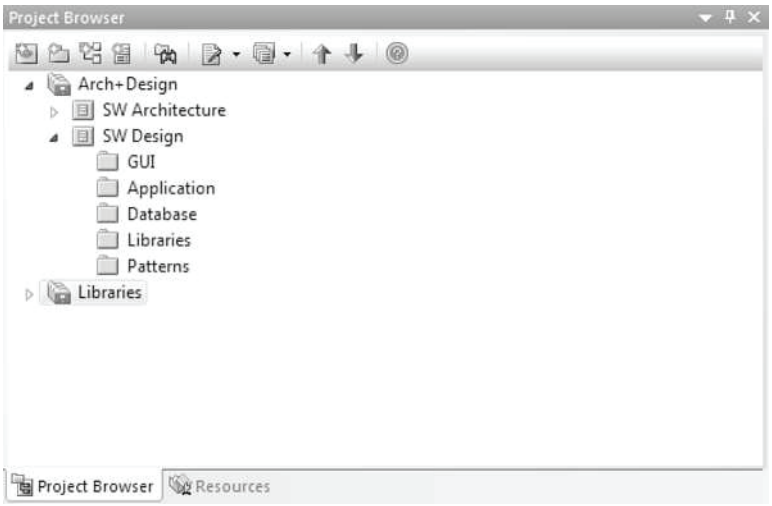

Abb. 3: Beispiel der Struktur im Enterprise Architect

Ein Diagramm aus dem Design kann in der Architektur referenziert sein. Auch innerhalb eines Modells, wie z.B. Design, kann ein Element wie ein Objekt als Repräsentanz einer Variablen in diversen Diagrammen referenziert sein.
Eine Umbenennung in Enterprise Architect wird sofort an allen referenzierenden Stellen sichtbar und wirksam. Ebenso führt das Löschen eines Elements aus dem Projekt dazu, dass an allen referenzierenden Stellen diese Referenz nicht mehr sichtbar ist. Das Löschen eines Elements aus einem Diagramm führt nicht zum Löschen des Elements aus dem ganzen Projekt.

Organisation der Referenzen
Jedes Element im Modell-Repository, sei es ein Diagramm oder eine Klasse, besitzt eine eindeutige Identifikationsnummer namens Globally Unique Identifier (GUID) .
Ein Diagramm oder ein Modell setzt sich zusammen, in dem es aus dem Pool an definierten Elementen diese per GUID referenziert. Einige Elemente sind durch Ihre Abhängigkeit von Start-/Zielelement indirekt definiert und besitzen keine GUID. Daher können diese Elemente nicht direkt per GUID referenziert werden.

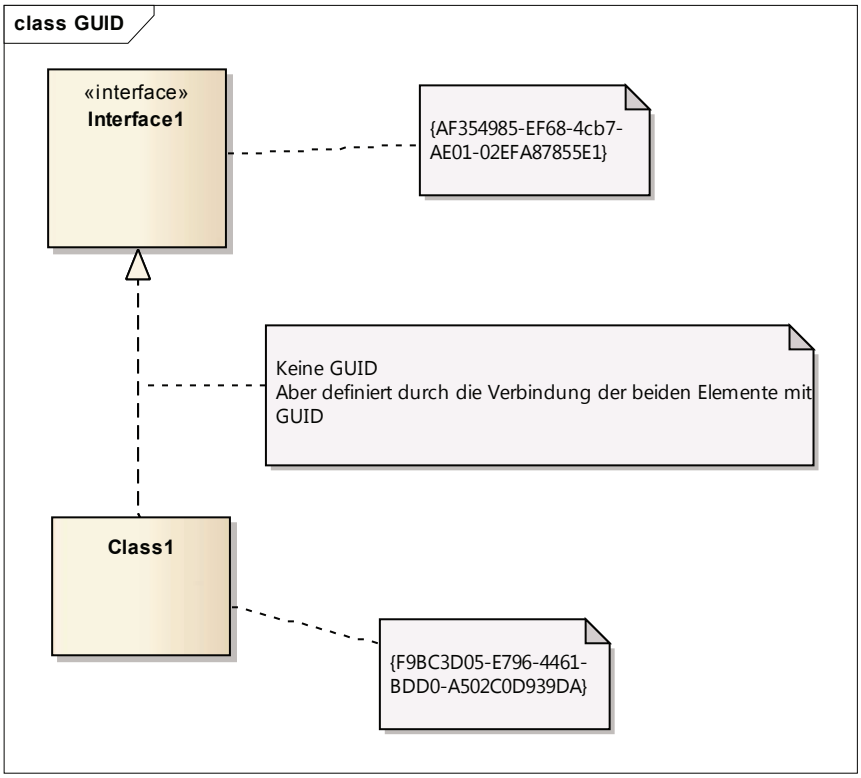

Abb. 4: Beispiel Abhängigkeit GUID

Eine GUID darf in einem Projekt nur für ein Element vorhanden sein. Diese Regel ist auf den ersten Blick selbstverständlich, stellt doch die GUID einen Schlüssel dar, wie er in jeder Datenbank auch nur einmal definiert werden darf.
Auf den zweiten Blick aber birgt genau diese Regel in der Arbeit mit Variationen, wie sie im Entwurf von Architektur und Design vorkommen können, ein Problem in sich. Auch die Arbeit mit Varianten wird dadurch beeinflusst.

Referenz zwischen Elementen aus verschiedenen Packages
Traceability wird dadurch geschaffen, wenn zwischen Softwarekomponenten, Architekturkomponenten und den Anforderungen eine Verbindung geschaffen wird. Diese Verbindung ist im Modell eine Referenzierung von vorhandenen Modellelementen. Somit werden Referenzen von Elementen aus unterschiedlichen Packages zueinander hergestellt.
Im Modell merkt sich diese Information immer das Startelement der Verbindung. Dieses Startelement ergibt sich aus der Richtung der Assoziation.

Szenarien in der Arbeit mit Enterprise Architect
Die nachfolgend dargestellten Möglichkeiten berücksichtigen die Arbeitsweise des Werkzeugs.

1. Varianten

In einem Projekt kann zu einer Zeit nur eine Variante geladen sein. Würde versucht werden, beide Versionen (Varianten) nebeneinander ins Modell zu laden, würde es vom Werkzeug eine Fehlermeldung geben.

Dies gilt deshalb, weil beide Versionen, obwohl in unterschiedlichen Packages, Elemente mit gleicher GUID enthalten. In einem Projekt darf es aber nur Elemente geben, die unterschiedliche GUID besitzen. Eine Umbenennung der Elemente hilft hier nicht.

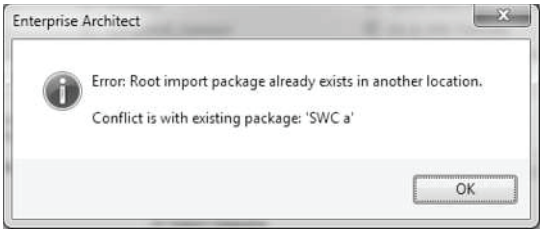

Abb. 5: Beispiel Fehlermeldung, wenn 2 Elemente gleiche GUID aufweisen sollten

Sollen mehrere Versionen/Varianten eines Modellteils in einem Modell geladen werden, bietet das Werkzeug an, die bestehende GUID durch eine neue GUID zu ersetzen (Strip GUID), um den Konflikt aufzulösen. Dadurch geht aber der Bezug zu dem Original/Master verloren. Ansonsten bleibt nur der Weg, ein weiteres Projekt anzulegen, um eine bestimmte Variante darin zu laden.

2. Design

Im Design werden Softwarekomponenten und dazugehörige Artefakte, wie z.B. Diagramme, zwischen den Projekten wiederverwendet. Sei es um diese so in gleicher Form zu benutzen oder um basierend darauf eine neue Version/Variante zu erstellen. Im Design oder in der Architektur wird gerne mit Referenzen auf andere Elemente oder Diagramme gearbeitet. Werden im Projekt Teile wiederverwendet, welche schon in einem anderen Projekt miteinander eine Beziehung hatten, so erscheint diese Beziehung sofort auch im neuen Projekt. Ansonsten muss eine Beziehung wieder erstellt werden.

3. Architektur

In der Architekturbeschreibung wird ähnlich wie im Design die Beziehung der Elemente/Diagramme untereinander gepflegt. Ein Architekturmodell kann natürlich auch wiederverwendet werden.
In der Architektur werden alternative Abläufe/Ansätze in der Dokumentation gefordert. Aus diesen Alternativen muss in der Regel die für das Projekt gewählte Variante begründet gewählt werden. Diese Alternativen sind alle in ein und demselben Projekt/Modell enthalten. Damit ist schon klar, dass es sich um „Kopien“ zwischen den einzelnen Alternativen handeln muss. Wenn man jede Alternative gesondert modelliert, würde es ausreichen, alle Alternativen in einem Diagramm darzustellen.

Lösungswege
Die parallele Darstellung von allen Varianten in einem Modell würde es erlauben Änderungen in einer Variante zu modellieren, ohne die anderen Varianten zu beeinflussen.
Dazu müsste es möglich sein, über eine neue Beziehungstabelle den Zusammenhang zwischen zwei Varianten mit Elementen gleicher GUID beizubehalten. Will man die Varianten nicht weiter modifizieren würde es ausreichen, ohne Beziehungstabelle zu arbeiten.

Unterstützung durch Erweiterungen des Werkzeugs
Enterprise Architect bietet Schnittstellen, um die Modelldaten in der Datenbank zu verändern oder per API Kommandos auf dem Werkzeug auszuführen.
Entsprechend gibt es eine Vielzahl von Plug-Ins, die sich das zu Nutze machen.

Empfehlungen
In der Arbeit mit Enterprise Architect im Zusammenhang mit Produktlinien/ Varianten haben sich einige Regeln bewährt.

Dont's
1. Nicht auf Kopien eines Projektes arbeiten.
2. Gleichzeitige Bearbeitung eines Packages mit seinem Inhalt durch verschiedene Personen

Do's
1. Konzept für Management der Modellteile
2. Verwendung eines Werkzeugs für Versionsmanagement
3. Verantwortliche(n) für Integration/Wartung definieren

Diese Regeln sollten nicht nur für das Werkzeug Enterprise Architect anwendbar sein, sondern sollten auch für andere Werkzeuge herangezogen werden können.

Zusammenfassung
Die UML bietet im Prinzip alles Notwendige für die Softwareentwicklung. Die Darstellung von Architektur, Design und deren Zusammenhänge mit Hilfe der UML sind in der Praxis schnell erstellt.
Leider bietet die UML wenig an definierter oder gar intuitiver Unterstützung für das Handling von varianten Modellteilen. Die Lösung dafür liegt auf Seiten der Entwicklungsmethodik und beim Werkzeug selber.
Am Beispiel des Werkzeugs Enterprise Architect wird gezeigt, wie sich der Vorteil der Verknüpfung zwischen Architektur und Design bei Hinzunahme von Varianten und Aspekten der Versionierung gleich in einem Komplexitätsgewinn auswirkt.
Diese Problematik liegt aber nicht in dem Werkzeug Enterprise Architect begründet, sondern stammt aus der grundlegenden Arbeitsweise solcher Werkzeuge.
Neben kommerziellen Erweiterungen können auch mit eigenen Vorgehensmodellen diese Hürden in der Praxis umgangen werden.

Literatur- und Quellenverzeichnis

[1] Sparx Systems Inc., www.sparx-systems.com.au
[2] V-Modell XT, http://www.v-modell-xt.de/
[3] Unified Process, http://en.wikipedia.org/wiki/Unified_Process
[4] arc42, www.arc42.de
[5] M. Anastasopoulos, C. Gacek, Implementing Product Line Variabilities, Fraunhofer Institute for Experimental Software Engineering (IESE)
[6] C. Atkinson et al, Component-based Product Line Engineering with UML, Addison-Wesley, August 2001

Autor

Dipl.-Ing. (FH) Jürgen Hartung leitet seit 2008 den Projektbereich Realtime Systems & Technologie-Transfer. Seit 1996 arbeitete er zuerst etliche Jahre als Entwickler und später als Berater bei Kunden im Automotive Umfeld. Schwerpunkte seiner fachlichen Tätigkeit liegen in der Einführung und Beratung zu Methoden der Spezifikation von Software-Architektur und Design. Daneben ist das Fachgebiet der Fahrzeugdiagnose der zweite Schwerpunkt.

Kontakt

Internet: www.koelsch-altmann.de
Email: juergen.hartung@koelsch-altmann.de