Was ist Software-Architektur?
Ein Abgleich mit der Praxis

Dominikus Herzberg
Studiengang Software Engineering
Hochschule Heilbronn
74081 Heilbronn

herzberg@hs-heilbronn.de

Abstract: Einige der gängigen Auffassungen zu „Was ist Software-Architektur?“ stimmen nicht überein mit Beobachtungen aus der industriellen Praxis der Software-Entwicklung – so die These dieses Papiers. Daraus wird ein Verständnis von Software-Architektur hergeleitet, das gesamtorganisatorische Zusammenhänge herstellt und, was neu ist, den ökonomischen Aspekt hervorhebt und für ein umfassenderes Verständnis wirbt.

1 Einleitung

Das Thema „Software-Architektur“ findet in jüngster Zeit auffällig Beachtung im deutschsprachigen Raum. Das macht die Anzahl der deutschsprachigen Bücher deutlich, die in den letzten drei, vier Jahren zum Thema erschienen sind, z.B. [PBG04, Sie04, Sta05, VAC+05]. Damit prägt sich ein europäisch geprägtes Meinungsprofil heraus, das durchaus vieles von den Veröffentlichungen aus Übersee übernimmt – berechtigterweise, da dort gute Forschung geleistet wurde. Aber es zeigt sich auch eine neue Reflektion über das Wesen und die Bedeutung von Software-Architektur, mit neuen, eigenen Antworten. Allerdings scheinen einige Antworten fernab von der industriellen Realität zu sein, so die These dieses Beitrags. In der Praxis hat der Autor ein anderes Begriffsverständnis von Software-Architektur kennen gelernt, das hier kurz skizziert und zur Diskussion gestellt werden soll.

Die meisten der deutschsprachigen Publikationen schließen sich mit geringen Abweichungen der Definition von [BCK03] an:

> The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

Die weitere Einordnung von Software-Architektur (SWA) in den Entwicklungsprozess führt zu Aussagen wie: Die Architektur bildet die „Brücke“ zwischen Anforderungen und Implementierung; SWA hilft vor der Implementierung Aussagen über Qualitätsmerkmale

zu treffen; sie hilft Folgen von Änderungen abzuschätzen; SWA bildet die frühe Möglichkeit zu verifizieren, ob Anforderungen erfüllbar sind oder nicht; SWA dient als Kommunikationsmedium zwischen den Stakeholdern. Dies und weiteres ist so oder ähnlich in verschiedenen deutschsprachigen Publikationen zu lesen.

Stimmt das mit den Erfahrungen aus der Praxis überein? Fehlen der Definition nicht wesentliche Elemente? Vertun wir hier die Chance, den Ansichten aus Übersee ein präziseres und umfassenderes Verständnis von Software-Architektur entgegen zu setzen? Das folgende Kapitel, Kapitel 2, schildert exemplarisch ein paar Beobachtungen aus der Praxis im Umgang mit Architektur. Kapitel 3 leitet daraus einen Vorschlag ab für ein „neues" Architekturverständnis und stellt die Konsequenzen eines solchen Verständnisses dar.

2 Beobachtungen aus der Praxis

Neue Anforderungen sollen analysiert werden. Gibt die SWA Auskunft darüber, ob sich die Anforderungen nahtlos in das System einbauen lassen und wie „teuer" die Umsetzung sein wird? Mitnichten! Die SWA ist viel zu grob dafür und gibt über die benötigten Informationen keine Auskunft. Ein Blick in die Praxis zeigt das Vorgehen eines großen Konzerns für Telekommunikationssysteme bei der Software-Entwicklung von Vermittlungsstellen (*switching systems*) für Mobilfunksysteme, siehe Bild 1. Es handelt sich dabei um die stete Fortschreibung von Legacy-Software. Bevor neue Anforderungen (*Requirement Specifications, RSs*) formuliert werden und den Ausgangspunkt für ein Projekt begründen (*Tollgate 0*), werden *Quick Studies* und *Scenario Studies* durchgeführt und durch Forschungsaktivitäten begleitet. Software-Architekten und Design-Experten sind in dieser frühen Phase der *Pre-PreStudy* involviert. Es werden hauptsächlich neuartige Fragestellungen, Konzepte und Entwicklungen und Konsequenzen aus der Standardisierung untersucht. Diese Phase dient einem Erfassen und dem Verständnis der Problemdomäne – die Software-Architektur bleibt davon unberührt. In der frühen Phase der *PreStudy* wird ausgehend von den Anforderungen direkt auf die Code/Design-Ebene abgestiegen und ein oder mehrere *Implementation Sketches* (IS) werden ausgearbeitet. Im Anschluß wird in der *Feasibility Study* eine Detaillierung durch *Implementation Proposals* (IP) vorgenommen. Die IPs bilden die Grundlage für sehr genaue Schätzungen der zu erwartenden Realisierungskosten. Und erst im Zusammenhang mit IS und IP beginnen die Dialoge mit und die Einbindung der Software-Architekten. Zusammen mit den Architekten wird entschieden, ob sich die skizzierten Lösungen in die SWA einfügen oder ob die SWA einer Änderung bedarf. Das Verständnis der SWA als Brücke zwischen Anforderungen und Implementierung will sich nicht einstellen. Ein Software-Architekt vermag mit Hilfe der SWA zwar noch zu entscheiden, welche Anforderungen vermutlich welche Teile des Systems betreffen werden und welche verantwortlichen Organsationseinheiten mit der *Feasibility Study* zu beauftragen sind – zu mehr taugt die Softwarearchitektur in dieser Hinsicht nicht. Zur Kostenabschätzung, wie erwähnt, erst recht nicht. Die Software-Architektur entwickelt sich stets in intensiver Auseinandersetzung mit dem Code und den Anforderungen, sie ist jedoch in erster Linie ein Abbild einer Organisationsstruktur zur Arbeitsteilung.

Ist SWA ein Kommunikationsmedium zwischen Stakeholdern? Ein Vergleich: Würde es

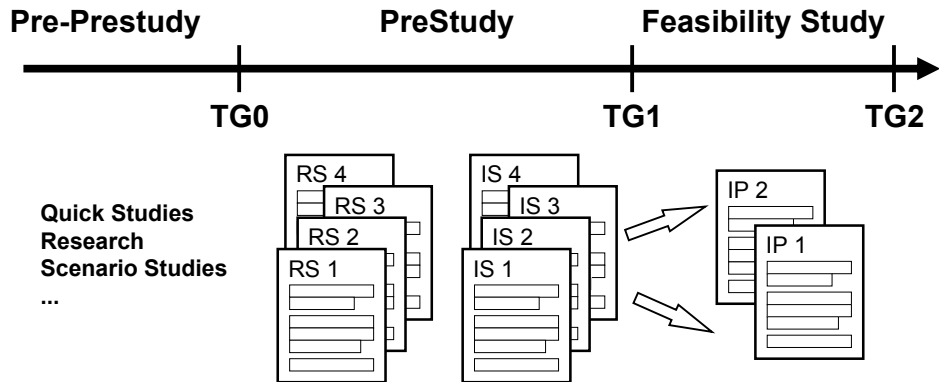

Abbildung 1: Architektur als Brücke zwischen Anforderungen und Implementierung? Der Prozess der frühen Phasen offenbart ein anderes Bild.

ein Stakeholder im Flugzeugbau wagen, mit einem kurzen Blick auf die Blaupausen an der Architektur eines Flugzeugs Kritik zu üben? Und das, obwohl er oder sie keine Ahnung von Aerodynamik und dem Know-How hat, die den Aufbau des Rumpfes, die Lage und Gestalt der Flügel und Triebwerke nicht beliebig sein lassen? Natürlich nicht. Jede Interessenspartei schickt stellvertretend kompetente Experten ins Feld. Das ist insbesondere bei komplexen, stark technisch-orientierten Softwaresystemen (Stichwort: embedded Systems) belegbar: In solchen Projekten stellen selbst Kunden oftmals hochspezialisierte Experten zur Projektbegleitung ab. Erst dann greift die Bedeutung der SWA für ein Softwaresystem in seiner Rolle als Kommunikationsmedium. Aber es ist und bleibt so, die Experten müssen das unter sich ausmachen. Was wohl richtig ist – und das ist wiederum eine Beobachtung aus der Praxis: den Stakeholdern muss die SWA für bestimmte Zwecke verständlich aufbereitet werden. Manager brauchen das, um Entscheidungen zu treffen, Marketing-Menschen, um Kunden eine Grobstruktur „verkaufen“ und die Vorzüge und Besonderheiten eines Systems darzustellen zu können. Diese Aufbereitungen für spezielle Zielgruppen werden zwar gerne „Architektur“ genannt, angemessen ist jedoch eher der Begriff der „Pseudo-Architektur“; der Begriff findet sich z.B. bei [VAC+05] und meint die weitgehende Entfernung technischer Details aus einer Architekturdarstellung. Es ist eine besondere Fähigkeit von Software-Architekten, die SWA für bestimmte Zielgruppen vereinfacht und verständlich aufzubereiten, ohne dabei die tatsächlichen Verhältnisse zu arg zu verzerren. In der Literatur wird diese Fähigkeit wenig gewürdigt und bestenfalls am Rande erwähnt.

Arbeitet man mit Software-Architekten zusammen, dann wird man feststellen, dass diese Menschen zwar zum Teil mit sehr einfachen Diagrammen arbeiten, sich dahinter aber ein immenses Fach- und Detailwissen verbirgt, das selbst eine Architektur-Dokumentation nie vollständig abzubilden vermag. Daran zeigt sich, was weiter oben beschrieben wurde: SWA ist undenkbar ohne teils intime Design- bzw. Programmkenntnisse. Ein Beispiel zeigt Abbildung 2a). Es ist die Layer 3-Architektur im GSM-System (Global System for Mobile communication) – ein kaum komplex wirkendes Bild, das im Standard vorgegeben

ist. Architekten in der Telekommunikation werden jedoch darauf bestehen, dieses Bild als Architektur zu bezeichnen. In der Tat können sich an einem solchen Bild stundenlange Diskussionen entzünden. Dabei haben die Architekten die nächste Zerlegungsstufe, die Architektur der konkreten Implementierung des Standards im Kopf, siehe Abbildung 2b), wieder ein Beispiel aus der Praxis. Die konkrete Architektur ist abbildbar auf die Vorlage des Standards, auch wenn das nicht direkt sichtbar erscheint. Hinter diesem „Kasten/Strich"-Diagramm „sehen" die Architekten Datenflüsse, Protokolle, Funktionen und Dienste. Solche Details sind meist ausschnitthaft und oft immer noch vergröbernd dokumentiert.

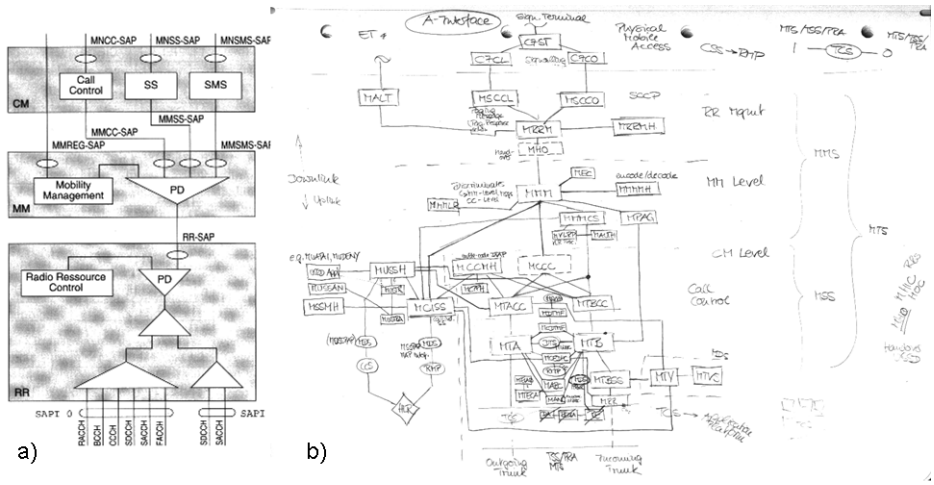

Abbildung 2: a) Die Layer 3-Architektur im GSM-System und b) die dazu gehörige Architektur einer konkreten Umsetzung.

Dennoch bilden Diagramme wie in Abbildung 2 und die dazugehörigen Beschreibungen die Eckpfeiler für die gesamte Konstruktion eines Softwaresystems. Die SWA markiert Einschränkungen und Freiheitsgrade. Was darf von den Designern auf „unteren" Ebenen frei verhandelt und festgelegt werden, was darf nicht ohne Einbindung der Architekten verändert werden? Die Architektur entzündet sich am Detail, um im Größeren Grenzen zu ziehen. Eine gute SWA und insbesondere gute SWA-Diagramme beweisen sich in der Praxis daran, inwieweit Architekten daran Einstiegspunkte finden, um wesentliche Sachverhalte konsistent und ausführlich diskutieren, effizient festschreiben und effektiv kommunizieren zu können.

3 Ein Vorschlag

Wie lässt sich das beobachtete Verständnis von SWA einfach abbilden und in einen Kontext einbinden? Bild 3 unterscheidet die an ein System adressierten Anforderungen von der

Implementierung. Das Bild vereinfacht die Zusammenhänge; die Pfeile zeigen an, was was beeinflusst. Bei allen Einflüssen gilt für die Implementierung das Kriterium, dass sie die an sie adressierten Anforderungen *erfüllen* soll.

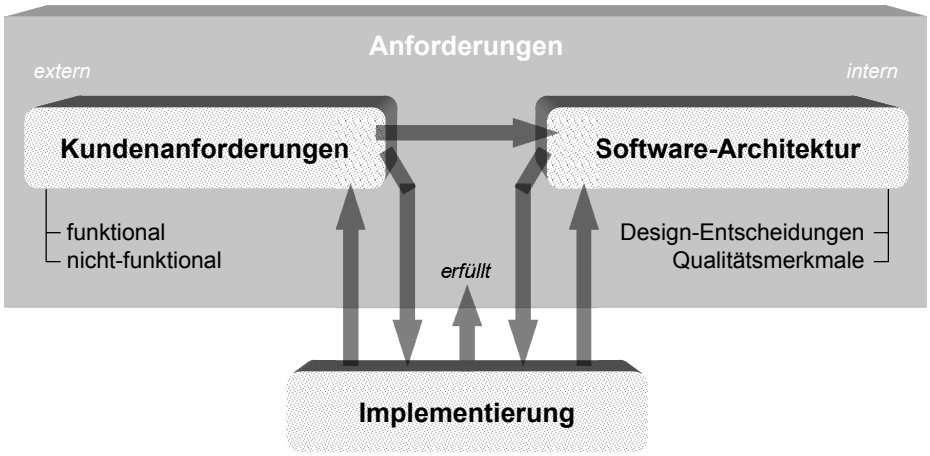

Abbildung 3: Software-Architektur als Ausdruck „interner“ Anforderungen

Bei den Anforderungen unterscheiden wir zwischen den *externen* Anforderungen und den *internen* Anforderungen. Die externen Anforderungen sind die Kundenanforderungen, funktionale wie nicht-funktionale. Dies sind die Anforderungen, für die ein Kunde zahlt, deren Erfüllung ihm eine gewünschte Dienstleistung oder ein Produkt realisiert. Den Kunden interessiert es (in den meisten Fällen) wenig, ob ein System von seinem Hersteller so oder anders strukturiert wird, ob es zukünftig leicht sein wird, Erweiterungen einzubauen oder nicht.

Diese Darstellung überzeichnet, hilft aber bei der Klärung und Einordnung von Architektur. Es ist Aufgabe und Anliegen der entwickelnden Organisation, sich darum zu kümmern, den Wert, den eine Softwareleistung oder ein -produkt für sie als Handelswert darstellt, zu erhalten, ihn zu pflegen und sich selbst gestellten Anforderungen zu unterwerfen. Diese internen Anforderungen repräsentieren die Software-Architektur. Einerseits sind dies Design-Entscheidungen, die die Grenzen für den weiteren Entwurf ziehen bzw. Freiheitsgrade zulassen, andererseits intern formulierte Qualitätsmerkmale (z.B. Erweiterbarkeit, Zuverlässigkeit, Fehlertoleranz). Das Entscheidende ist, dass eine entwickelnde Organisation diese „inneren“ Anforderungen, die Software-Architektur, selbst definiert und sich selbst gegenüber in der Implementierung einfordert. Das geschieht selbstverständlich auch als Reaktion auf externe Anforderungen und dem durchaus verständlichen Anliegen des Kunden auf „Werterhaltung“ einer bezogenen Leistung bzw. eines bezogenen Produkts. Das kann soweit gehen, dass ein Kunde auf die Offenlegung solcher intern gestellten Anforderungen besteht. Den Einfluss der Software-Architektur auf die Kundenanforderungen halte ich für eher gering.

Dieses Verständnis von Architektur wird einer umfassenderen und praxis-orientierten Auf-

fassung eher gerecht. Es bettete SWA in einen gesamtorganisatorischen Zusammenhang ein und erweitert den Raum dessen, was Architektur ausmacht. Sie ist nicht nur die Struktur von Software-Elementen eines System. Es beraubt die SWA um die vermeintliche Funktion als Brücke, rückt sie aber andererseits gleichermaßen in den Mittelpunkt wie Kundenanforderungen. Die Architektur entsteht wesentlich durch die Auseinandersetzung mit der Implementierung, dem Code. Der Code lässt sich nicht in beliebige Form unter beliebigen Vorgaben pressen – manche interne wie externe Anforderung wird ansonsten unerfüllbar. So reift mit der Implementierung und der Systemevolution auch die Architektur.

In diesem Sinne ist die oben zitierte Definition zur SWA zu eng, zumal ihr ein wesentlicher Zusammenhang abgeht: der ökonomische Aspekt der SWA. Die Beschreibung von Software-Architektur folgt dem ökonomischen Prinzip, wie bedeutsam solche internen Anforderungen für das SW-Produkt sind, welcher Grad und welcher Umfang ausreichen. Was lebt in den Köpfen einer Organisation, was bedarf der ausdrücklichen Dokumentation? Welcher Aufwand ist gerechtfertigt? Das erklärt, warum sich in Unternehmen mehr oder weniger hochwertige Architekturbeschreibungen finden. Die Qualität und der Umfang einer SWA-Beschreibung ist das Produkt eines ökonomischen Prozesses, der die internen Anforderungen, die Software-Architektur, als *Asset* begreift.

Literatur

[BCK03] Len Bass, Paul Clements und Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, Boston, 2. Auflage, 2003.

[PBG04] Torsten Posch, Klaus Birken und Michael Gerdom. *Basiswissen Softwarearchitektur: Verstehen, entwerfen, bewerten und dokumentieren*. dpunkt, Heidelberg, 2004.

[Sie04] Johannes Siedersleben. *Moderne Software-Architektur: Umsichtig planen, robust bauen mit Quasar*. dpunkt, Heidelberg, 2004.

[Sta05] Gernot Starke. *Effektive Software-Architekturen: Ein praktischer Leitfaden*. Hanser, 2. Auflage, 2005.

[VAC+05] Oliver Vogel, Ingo Arnold, Arif Chughtai, Edmund Ihler, Uwe Mehlig, Thomas Neumann, Markus Völter und Uwe Zdun. *Software-Architektur: Grundlagen, Konzepte, Praxis*. Spektrum Akademischer Verlag, 2005.