Vorwort

Worüber reden wir, wenn wir uns über »Architektur« im Allgemeinen unterhalten?

Wie bei allen metaphorischen Annäherungen kann auch die Betrachtung einer Software unter architektonischen Gesichtspunkten gleichermaßen viel verhüllen wie offenbaren. Ebenso kann sie mehr versprechen, als sie zu liefern imstande ist, oder mehr liefern, als sie ursprünglich zu versprechen schien.

Der offenkundige Reiz der Architektur besteht in ihrer Struktur, deshalb steht Letztere auch im Bereich der Softwareentwicklung im Fokus sämtlicher Paradigmen und Überlegungen – Komponenten, Klassen, Funktionen, Module, Layer und Services, egal, ob Mikro oder Makro. Allerdings erweist sich die Gesamtstruktur vieler Softwaresysteme nicht selten als fragwürdig oder widersinnig – etwa wie die sowjetischen Kollektivbetriebe, die als Vermächtnis für das Volk bestimmt waren, undenkbare Jenga-Türme, die bis zum Himmel hinaufragen, oder wundersame, unter riesigen Schlammlawinen begrabene archäologische Fundschichten. Dass Softwarestrukturen in der gleichen Art und Weise unserer Intuition folgen, wie dies auch bei Gebäudestrukturen der Fall ist, lässt sich häufig nur schwer erkennen.

Gebäude besitzen dagegen eine unübersehbare physische Struktur – ob in Stein oder Beton verwurzelt, ob hoch nach oben ragend oder weit in die Breite verlaufend, ob groß oder klein, ob atemberaubend oder schlicht und banal. Bei Bauwerken gibt es kaum eine Alternative, als sie nach der Physik der Schwerkraft und den verwendeten Materialien auszurichten. Im Gegensatz dazu hat Software – außer im Sinne der Realitätstreue – wenig für die Schwerkraft übrig. Und woraus besteht Software? Anders als Gebäude, die aus Ziegeln, Beton, Holz, Stahl und Glas gefertigt werden, ist Software eben nur aus Software gemacht. Große Softwarekonstrukte setzen sich aus kleineren Softwarekomponenten zusammen, die wiederum aus noch kleineren Softwarekomponenten bestehen und so weiter, und so fort. Oder, wie es Stephen W. Hawking in seinem Buch »Eine kurze Geschichte der Zeit« ausdrückt:

> *Da stehen lauter Schildkröten aufeinander.*[1]

Wenn wir über Softwarearchitektur im Speziellen reden, geht es darum, dass Software in ihrer Beschaffenheit rekursiv und fraktal, im Code prägnant und richtung-

1 Stephen W. Hawking, *Eine kurze Geschichte der Zeit*, Rowohlt Verlag, 2004.

weisend ist. Einfach alles hat mit Details zu tun. Zwar spielen ineinandergreifende Detailebenen auch bei der Architektur von Gebäuden eine Rolle, in Bezug auf Software ist es allerdings kaum sinnvoll, über physische Maßstäbe nachzudenken. Software besitzt eine Struktur – eigentlich jede Menge und zahlreiche Arten von Strukturen –, deren Vielfalt das Spektrum der physischen Gebäudestrukturen problemlos in den Schatten stellt. Man kann durchaus behaupten, dass dem Design in der Softwareentwicklung mehr Einsatz und Aufmerksamkeit zuteilwird, als dies in der Gebäudearchitektur der Fall ist – und insofern ist es auch nicht unbedingt abwegig, die Softwarearchitektur als architektonischer zu betrachten als die Gebäudearchitektur!

Aber physische Maßstäbe sind für den menschlichen Verstand besser zu begreifen. Sie bieten uns Orientierung in der Welt, deshalb halten wir stets Ausschau danach. Und sicherlich sind die einzelnen Kästen in schematischen PowerPoint-Darstellungen ansprechend und vermitteln einen klaren visuellen Eindruck, trotzdem geben sie nicht den vollen und lückenlosen Umfang der Architektur eines Softwaresystems wider. Zweifellos bieten sie eine bestimmte Sicht auf einen architektonischen Aufbau; die Kästen allerdings fälschlicherweise mit dem großen Ganzen – also der Architektur selbst – zu verwechseln, heißt definitiv, das große Ganze – und somit die Architektur als solche – aus den Augen zu verlieren: Softwarearchitektur sieht nicht irgendwie besonders aus. Eine spezifische Visualisierung ist eine Entscheidungsfrage, keine Gegebenheit. Vielmehr handelt es sich um eine Entscheidung, die auf einer weiteren Auswahl von Möglichkeiten basiert, nämlich: was enthalten sein soll, was durch Form oder Farbgebung hervorgehoben werden soll, was durch Gleichförmigkeit oder Auslassung heruntergespielt werden soll. Eine Sichtweise hat gegenüber einer anderen nichts Natürliches oder Intrinsisches an sich.

Nun mag es nicht viel Sinn machen, sich im Kontext der Softwarearchitektur mit physikalischen Gesetzmäßigkeiten und physischen Maßstäben auseinanderzusetzen, dennoch müssen wir auch in diesem Bereich bestimmte physische Einschränkungen bedenken und entsprechend berücksichtigen. Prozessorgeschwindigkeiten und Netzwerkbandbreiten können ein hartes Urteil über die Performance eines Systems fällen. RAM- und Datenspeicherkapazitäten können den Ambitionen jeder Codebasis Grenzen setzen. Software mag einer dieser Stoffe sein, aus denen Träume gemacht sind, sie wird aber trotz allem in einer physischen Welt betrieben.

> *Das ist das Ungheuerliche in der Liebe, dass der Wille unendlich ist und die Ausführung beschränkt, dass das Verlangen grenzenlos ist und die Tat ein Sklave der Beschränkung.*
>
> – William Shakespeare[2]

2 Shakespeare, *Troilus und Cressida*, um 1601, Erstdruck 1610, erste deutsche Übers. von Johann Joachim Eschenburg, 1777. Hier übersetzt von Wolf Graf Baudissin, Georg Andreas Reimer, Berlin, 1832.

Es ist die physische Welt, in der wir ebenso wie unsere Unternehmen und unsere Volkswirtschaften existieren. Und diese Tatsache liefert uns einen weiteren Kalibrierungsgesichtspunkt, anhand dessen wir die Softwarearchitektur verstehen können – andere, weniger physische Kräfte und Größen, auf deren Grundlage wir uns verständigen und argumentieren können.

> *Architektur repräsentiert die signifikanten Designentscheidungen, die ein System formen und gestalten, wobei die Signifikanz an den Kosten von Änderungen bemessen wird.*
>
> – Grady Booch

Zeit, Geld und Aufwand geben uns einen Maßstab vor, um das Große und das Kleine, das Wesentliche und das weniger Wesentliche zu sortieren und die architektonischen Aspekte von dem Rest zu unterscheiden. Dieser Maßstab sagt uns auch, wie wir feststellen können, ob eine Architektur gut ist oder nicht: Eine gute Softwarearchitektur erfüllt die Bedürfnisse aller User, Entwickler und Product Owner (Produkteigentümer), und zwar nicht nur zu einem gegebenen Zeitpunkt, sondern langfristig.

> *Wenn Sie denken, eine gute Architektur sei teuer, dann probieren Sie es mal mit einer schlechten.*
>
> – Brian Foote und Joseph Yoder

Die Modifikationen und Anpassungen, die im Rahmen einer Systementwicklung typischerweise vorzunehmen sind, sollten nicht von der Art sein, dass sie kostspielig und schwer zu realisieren sind und eine jeweils eigene Projektverwaltung erforderlich machen, sondern in die täglichen und wöchentlichen Arbeitsabläufe eingebunden werden können.

Und das bringt uns zu einem nicht unerheblichen Physik-orientierten Problem: der Zeitreise. Wie wissen wir, welcher Art diese typischen Modifikationen bzw. Anpassungen sein werden, sodass wir die damit einhergehenden signifikanten Entscheidungen darauf ausrichten können? Wie reduzieren wir den zukünftigen Entwicklungsaufwand und die Kosten, ohne Kristallkugeln und Zeitmaschinen zu Hilfe zu nehmen?

> *Architektur ist die Menge der Entscheidungen, von denen Sie wünschten, dass Sie sie bereits frühzeitig in einem Projekt richtig treffen, bei denen die Wahrscheinlichkeit, sie auch tatsächlich richtig zu fällen, aber nicht unbedingt höher ist als bei allen anderen Entscheidungen auch.*
>
> – Ralph Johnson

Das Vergangene zu verstehen, ist schon schwierig genug. Unser Verständnis von dem Gegenwärtigen ist bestenfalls vage – und die Vorhersage des Zukünftigen ist alles andere als trivial.

An diesem Punkt verzweigt der Weg in viele Richtungen.

Auf dem dunkelsten Pfad wartet die Vorstellung, dass starke und stabile Architekturen direkte Abkömmlinge von Autorität und Starrheit sind. Ist eine Modifikation kostenintensiv, wird sie verworfen, die ursächlichen Beweggründe werden kleingeredet oder vollständig in bürokratischen Abgründen versenkt. Das Mandat des Architekten ist total und totalitär – und als Folge davon verkümmert die Architektur zu einer Dystopie für ihre Entwickler und eine ständige Quelle der Frustration für alle anderen.

Entlang eines anderen Abzweigs richtet sich das Interesse hingegen auf die sauberste Variante. Hier wird der »Weichheit« der Software Rechnung getragen und darauf hingearbeitet, sie als vorrangigste Eigenschaft des Systems zu bewahren. Ebenso wird nicht nur die Tatsache berücksichtigt, dass wir auf der Grundlage unvollständigen Wissens arbeiten, sondern auch anerkannt, dass dies für uns menschliche Wesen zudem etwas ist, worin wir gut sind. Diese Vorgehensweise kommt unseren Stärken mehr entgegen als unseren Schwächen. Wir erschaffen Dinge und wir entdecken Dinge. Wir stellen Fragen und führen Experimente durch. Eine gute Architektur kommt genau dann zustande, wenn wir sie als Reise und nicht als Ziel verstehen, mehr als einen fortlaufenden Prozess des Untersuchens denn als unumstößliches Artefakt.

> *Architektur ist eine Hypothese, die durch Implementierung und Bewertung bewiesen werden muss.*
>
> – Tom Gilb

Das Beschreiten dieses Pfades erfordert Sorgfalt und Aufmerksamkeit, Überlegungen und Beobachtung, Praxis und Prinzipien. Auf den ersten Blick mag sich dies nach einem schleppenden Prozess anhören, im Endeffekt hängt jedoch alles davon ab, wie Sie den Weg beschreiten.

> *Der einzige Weg, um schnell voranzukommen, ist gut voranzukommen.*
>
> – Robert C. Martin

Genießen Sie die Reise.

Kevlin Henney, Mai 2017