Video: Making Architecture Matter - Martin Fowler Keynote

Der Begriff „Software Architektur“ hat für den Sprecher etwas ungeschicktes. Der Begriff beschwört die Vorstellung einer älteren Person herauf, die in einem Unternehmen arbeitet und Regeln für Software schreibt - vielleicht aber seit Jahren schon selbst keine Software mehr geschrieben hat, und dies ist eher unglücklich. Code ist wichtig, und eine Person, die im technischen Bereich arbeitet, sollte es gewöhnt sein, regelmäßig zu programmieren. Mit der Zeit hat sich der Mythos ergeben, dass Architektur weiter geht, als nur das Programmieren, also sollten Software Architekten auch nicht mehr programmieren - aber dies ist falsch!

Aber was ist Architektur wirklich? Und was bedeutet der Begriff in der Software?
Wir haben uns den Begriff aus der Konstruktion von Gebäuden ausgeliehen, er bedeutet jedoch eigentlich etwas komplett anderes. Die aktuell gültige Definition ist:

„The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution“
(ANSI/IEEE Std 1471-2000)

Das Problem hier ist: Die Definition beruht auf der Vorstellung, dass das, was die höchste Komponente ist, auch die wichtigste ist.
Aber genauso wichtig sind auch deren Bedeutungen und Beziehungen.

———————

Das, was in der Software Architektur besonders wichtig ist, ist ein allgemeines, geteiltes Verständnis, wie das Projekt funktioniert, das ist die Architektur. Alles, was alle Programmierer und Entwickler zusammen von dem Projekt und Projektdesign verstehen ist das, was wichtig ist. Es ist also auch der soziale Aspekt der wichtig ist, nicht nur die Diagramme wie wir sie kennen und so weiter.
Wenn man also möchte, dass das Projekt wächst, dann muss man sichergehen, dass alle, die daran arbeiten (und vor allem auch die Projektleiter), ein gutes allgemeines Verständnis davon haben, um was es geht.

Ein weiterer wichtiger Teil der Definition sind die Design-Entscheidung, die früh gemacht werden müssen. Eigentlich aber sind hiermit die Entscheidungen gemeint, von denen man sich *wünscht,* dass man sie sehr früh richtig treffen kann, weil man für genau diese Entscheidungen meistens nicht alle nötigen Informationen vorliegen hat (meistens).
Das sind also die Entscheidungen, die schwierig wieder zu verändern sind, wie z.B. die gewählte Programmiersprache, etc.

Man hat also: 1. allgemeines, geteiltes Verständnis und 2. schwer zu verändernde Entscheidungen - das ergibt dann: das wichtige Zeugs, was auch immer das sein mag. Denn das erste, was man herausfinden muss ist eben, was überhaupt wichtig ist, denn dies ist der Schüssel.

Aber warum sollten wir uns überhaupt um Software Architektur kümmern?
Oft muss man sich zwischen Qualität und Quantität entscheiden. Der Punkt ist hier: Es gibt in der Software immer einen Teil, den Kunden sehen können, und einen Teil, den sie *nicht* sehen können -> es gibt also interne und externe Qualität.
Klar scheint die interne Qualität erstmal nicht so wichtig, denn man sieht sie ja nicht. Bei der Kaufentscheidung wird sie also erstmal auch nicht berücksichtigt.
Man stellt sich aber selbst Steine in den Weg, wenn man mit schlechter interner Qualität arbeitet, denn dies erweist sich mit voranschreitender Zeit als negativ.
Möchte man seine Software mit Features erweitern, so wirkt sich das bei schlechtem Design wenig auf die Funktionalität aus. Bei gutem Design allerdings kann man seine Software sehr gut erweitern und sie so funktional halten.
Es macht also einen ökonomischen Unterschied, ob man eine gute Software Architektur hat oder nicht, und das sogar in relativ kurzer Zeit.