YOU AIN'T GONNA NEED IT?

Kent Beck, einer der Mitbegründer des eXtreme Programming, wurde in den 90er Jahren auf einer Konferenz danach gefragt, was denn aus seiner Sicht Softwarearchitektur sei. Seine lapidare Antwort lautete: „Das, was Softwarearchitekten machen." „Und was ist ein Softwarearchitekt?", wurde nach einigem Kichern nachgehakt. „Softwarearchitekt ist ein pompöser Titel, den Programmierer auf ihrer Visitenkarte haben wollen, um ihre üppigen Bezüge zu rechtfertigen", war Becks provokante Antwort.

Deutlicher kann man eigentlich nicht zum Ausdruck bringen, auf welche Weise die rasante Verbreitung agiler Entwicklungsverfahren Druck auf die immer noch junge Disziplin der Softwarearchitektur ausübt. Und auch in der Praxis kommen gerade die Softwarearchitekten, die nicht in der täglichen Entwicklungsarbeit fest verankert sind, in zunehmende Erklärungsnot, warum die Entwicklungsteams die mit viel Aufwand entworfene Unternehmensarchitektur beachten sollten: „You ain't gonna need it", wird ihnen entgegnet und ein viel einfacherer Ansatz für die speziellen Anforderungen implementiert. Das Ergebnis ist ein oft nicht unerheblicher Widerstand der Architektengilde gegen die Einführung agiler Verfahren, der auf der berechtigten Sorge basiert, jetzt nicht vorschnell alle mühsam erarbeiteten Architekturgrundsätze über Bord zu werfen.

Es gibt verschiedene Ansätze, diesem Spannungsfeld seine negative Energie zu nehmen: Der erste ist: Die Welt ist nicht schwarz und nicht weiß, sondern besteht aus den vielen Grautönen dazwischen. **Matthias Ehlert** und **Ramon Anger** propagieren in ihrem Artikel „Änderbarkeit auf Lebenszeit: Lean-Prinzipien in der Architektur" einen praktikablen iterativen Mittelweg zwischen den beiden Ansätzen und liegen damit ziemlich im aktuellen Trend.

Viele Aufgaben des Softwarearchitekten werden aus diesem Anlass gerade verschlankt. **Katja Eisentraut** und ihre Kollegen stellen in ihrem Beitrag „Aggregation und Kompostierung: Was ist da nochmal der Unterschied?" mit dem „Model Sketching" einen leichtgewichtigen Modellierungsansatz vor. **Urs Enzler** hält ein flammendes Plädoyer für die Arbeit eines Softwarearchitekten in agilen Arbeitsumfeldern, mahnt aber auch: „Aber aufgepasst: Ein agiler Softwarearchitekt braucht neben dem Wissen über Trends und Technologien auch große Ohren, um den verschiedenen Stakeholdern gut zuhören zu können. Und gute Augen, um beim hektischen Alltag den Überblick nicht zu verlieren." In diesem Zusammenhang finde ich auch den Artikel von **Roland Mast** äußerst spannend, in dem er aus der Analogie zur Musik herleitet, dass moderne Softwarearchitekten sich in ihrem Selbstverständnis im Team eher an Jazz-Musikern orientieren sollten als an klassischen Symphonieorchestern.

Es gibt aber auch einen anderen Ansatz, wie sich die Spannungen auflösen lassen: Conway's Law besagt, dass die Struktur von Entwicklungsorganisationen die Struktur der von ihnen entworfenen Systeme bestimmt. Danach wäre es nur folgerichtig, dass bei konsequenter Implementierung einer agilen Organisation automatisch Architekturansätze entstehen, die vertikal geschnittene, rein funktional gegliederte Komponenten abbilden. **Stefan Kraus** und seine Koautoren stellen in ihrem Artikel „Teile und herrsche: Kleine Systeme für große Architekturen" ein beeindruckendes, real existierendes Beispiel einer solchen leichtgewichtigen Architektur vor und belegen damit einmal mehr die Gültigkeit von Conway's Law.

Also besinnen wir uns bei unserem Streben nach einer leichtgewichtigen Architektur einfach auf eine der modernen Definitionen einer Softwarearchitektur, der zufolge diese die Summe aller wichtigen Entwurfsentscheidungen ist. Und „wichtig" bedeutet dabei, dass die Rücknahme einer Entscheidung mit hohem Aufwand und Kosten verbunden ist. Die so definierte Architektur wird auf jeden Fall im agilen Sinne wenig Verschwendung beinhalten, wenn wir die Entscheidungen nach Kent Becks Vorschlag entwerfen: „Do the simplest thing that might possibly work."

Ihr Thorsten Janning