INTERVIEW MIT SIMON BROWN

„DevOps und Softwarearchitektur ergänzen sich“

Agil und Architektur stehen nicht im Widerspruch zueinander.

Softwarearchitektur scheint Ihre Leidenschaft zu sein. Heute regiert agil die Welt. Wie passen diese beiden Dinge zusammen?

Simon Brown: Die Softwareentwicklungs-Branche hat eine Reihe von Ansätzen für die detaillierte Planung und Modellierung versucht, bevor irgendwelcher Code geschrieben wird. Das führte zu Wasserfall-Methoden, in denen jedes einzelne Detail zum Projektbeginn entschieden wird. Wie wir gelernt haben, erlaubt dieser Up-Front-Designansatz keine Rückkopplungsschleifen und ist resistent gegen Veränderungen.

Leider sind jetzt viele Teams auf das genau entgegengesetzte Extrem umgesprungen und machen nur noch sehr wenig Up-Front-Design. Zum Teil kann das sicherlich auf die agile Bewegung zurückgeführt werden – obwohl es nicht richtig ist, dass agil kein Up-Front-Design verlangt. Aber das ist das, was jeder glaubt, wenn er das Wort Architektur hört.

Meiner Ansicht nach ist ein Mindestmaß an Up-Front-Design wichtig, um die wesentlichen architektonischen Entscheidungen verstehen und treffen zu können. Nur so kann die Struktur der zu bauenden Software erkannt und erstellt werden. Was sind die großen Bausteine und wie stehen sie miteinander in Beziehung?

Es sollten auch die größten Risiken identifiziert und abgemildert werden. Dieser Ausgangspunkt ist für jedes Softwareprojekt sinnvoll – unabhängig davon, ob es agil ist oder nicht. Darüber hinaus: Wer Agilität von seinem Softwaresystem erwartet, der muss sie hineinbauen.

Aus meiner Sicht zeichnet sich eine gute Architektur (zum Beispiel gut strukturiert, hohe Kohäsion, lose Kopplung et cetera) dadurch aus, dass sie Agilität ermöglicht und mit Veränderungen in der Umgebung klarkommt. Und eine solche Qualität bekommt man nicht einfach so.

Aber heute gibt es doch andere Herangehensweisen wie Test Driven Development (TDD) oder Behaviour Driven Development (BDD) …

Brown: Software wird immer komplexer und umfangreicher. Daher ist es heute wichtiger denn je, über Softwarearchitektur nachzudenken. Aus meiner Sicht ist Softwarearchitektur der Rahmen, in dem TDD stattfindet.

Mit anderen Worten: Erstellen Sie ein Up-Front-Design, um die wichtigsten architektonischen Entscheidungen (zum Beispiel Technologieentscheidungen, architektonischer Stil et cetera) treffen zu können und die grundlegenden strukturellen Bausteine zu verstehen. Erst dann verwenden Sie Techniken wie TDD, um das Design weiter voranzutreiben und letztlich das System zu implementieren.

Sie haben das C4-Softwarearchitektur-Modell entwickelt. Worum geht es da?

Brown: Das C4-Modell hilft, die statische Struktur eines Softwaresystems über verschiedene Abstraktionsebenen hinweg zu verstehen. Es geht darum, eine gemeinsame Sprache zu

Simon Brown
ist unabhängiger Consultant, der sich auf Softwarearchitektur spezialisiert hat. Er ist der Autor von „Software Architecture for Developers“ [1]. Außerdem ist er der Erfinder des C4-Softwarearchitektur-Modells und der Gründer von Structurizr, einer Sammlung von kommerziellen und Open-Source-Tools, die Softwareteams dabei helfen, Softwarearchitektur zu visualisieren, zu dokumentieren und erfahrbar zu machen. Simon Brown hält eine der drei Keynotes auf der Developer Week 2017 (www.developer-week.de), die vom 26. bis 29. Juni 2017 in Nürnberg stattfindet.

schaffen, die alle Teammitglieder in die Lage versetzt, Softwarearchitektur zu beschreiben und darin zu denken. Mit einer OO-Programmiersprache im Hinterkopf besteht ein Softwaresystem aus einem oder mehreren Containern (Webapplikationen, mobile Apps, Standalone-Applikationen, Datenbanken, Dateisysteme et cetera), die jeweils wieder eine oder mehrere Komponenten enthalten. Diese wiederum werden von einer oder mehreren Klassen implementiert.

Diese Hierarchie von strukturellen Bausteinen kann verwendet werden, um ein Softwaresystem zu beschreiben, wobei Diagramme auf jeder Ebene eingesetzt werden, um diese Beschreibung darzustellen.

Es ist eine statische Struktur und ein abstraktionsorientierter Ansatz, mit der Idee, dass Teams ihre eigene Kästchen-und-Linien-Notation verwenden, wenn sie diese Diagramme erstellen.

Was denken Sie: Wie viele Projekte scheitern, weil die Architektur nicht passt?

Brown: Natürlich gibt es Teams, die ihre Softwarearchitektur nicht sorgfältig genug auswählen. Als Ergebnis kommt etwas heraus, das für das Problem, das sie lösen wollen, unpassend ist. In seinem Buch „Just Enough Software Architecture“ [2] nennt George Fairbanks dieses Phänomen „Architecture-indifferent Design“.

Es gibt auch Teams, die sich aus den falschen Gründen heraus für eine bestimmte Architektur entscheiden. Ich denke, so etwas werden wir gerade im Zuge der Microservices mehr und mehr antreffen.

Und genau das ist der Grund, warum ein gewisses Maß an Up-Front-Design für die meisten Softwaresysteme essenziell ist: Es ist extrem wichtig, die Hauptknackpunkte in der Architektur verstanden zu haben – also funktionale Anforderungen, Qualitätsmerkmale und Umweltauflagen – und einen geeigneten Startpunkt festzulegen, um ihnen zu begegnen.

Um die schwerwiegendsten technischen Risiken zu erkennen und abzumildern, würde ich Teams auch raten, Techniken wie Prototyping oder das Walking Skeleton einzusetzen.

Stören neue Rollen wie DevOps die Softwarearchitektur?

Brown: Nein, ganz im Gegenteil! DevOps und Softwarearchitektur ergänzen sich. Viele der Techniken, die wir mit DevOps verbinden (wie die automatisierte, kontinuierliche Bereitstellung von Infrastruktur), erfordern ein solides Verständnis der Software, die gebaut werden soll. Darüber hinaus müssen Faktoren, die diese Techniken leichter machen (etwa „The Twelve-Factor App“ [4]), im Voraus bedacht und beim Up-Front-Design berücksichtigt werden – genau wie Sicherheitsaspekte und Anforderungen an die Skalierbarkeit. ■

[1] Software Architecture for Developers, www.dotnetpro.de/SL1706Interview1

[2] Just Enough Software Architecture, George H. Fairbanks, Herausgeber Marshall & Brainerd, ISBN-13: 9780984618101, www.dotnetpro.de/SL1706Interview2

[3] Walking Skeleton, www.dotnetpro.de/SL1706Interview3

[4] The Twelve-Factor App, https://12factor.net/de