

DIE ARCHITEKTURVISION IN SCRUM: VORAUSSPLANUNG UND EMER- GENTES DESIGN BALANCIEREN

Um sich ändernde Anforderungen bestmöglich zu unterstützen, muss die Architektur in der agilen Softwareentwicklung inkrementell von Sprint zu Sprint entstehen. Damit das Team aber nicht ständig alles umbauen muss, ist eine initiale Architekturvorstellung notwendig. Dieser Artikel stellt das Konzept der Architekturvision vor und beschreibt Kriterien und Techniken für ihre Erstellung.

Die Architekturvision im Überblick

Um ein Scrum-Projekt zielgerichtet zu starten, müssen wir nicht nur verstehen, wie das Produkt grob aussehen und was es grob leisten soll (vgl. [Sch04], [Pic10]). Wir sollten auch eine Vorstellung davon haben, wie wir die Software in etwa erstellen wollen und welche Architekturprinzipien und Technologien zum Einsatz kommen. Nur so können wir die Machbarkeit des Produkts beurteilen, einen realistischen Kosten- und Zeitrahmen ermitteln und ein schlagkräftiges Team zusammenstellen. Und nur so lässt sich das Produkt inkrementell entwickeln und architektonische Refaktorisierungsarbeiten lassen sich gering halten. Diese Vorstellung bezeichnen wir als *Architekturvision*¹⁾.

Die Architekturvision stellt eine zielgerichtete und nachhaltige Entwicklung der Software (*Sustainable Development*) sicher. Wie ihr Name andeutet, dokumentiert sie das gemeinsame Softwarearchitekturbild des Teams. Dabei sollte die Vision möglichst leichtgewichtig sein und lediglich die für den Produkterfolg kritischen Architektur Aspekte festlegen. Hierzu zählen insbesondere Bereiche, die sich während der Entwicklung nur mit hohen Kosten ändern lassen, wie der Architekturstil und die verwendeten Programmiersprachen. Alles andere sollte *nicht* in der Architekturvision beschrieben werden, denn die Architektur in Scrum wächst inkrementell und wird von Sprint zu Sprint verfeinert. Somit ist es nicht Aufgabe der Architekturvision, alle Fragen vorab zu klären.

¹⁾ Die Architekturvision ergänzt die Produktvision. Letztere beschreibt welches Produkt für wen und warum entwickelt werden soll. Vgl. auch [Pic10] für eine ausführliche Diskussion der Produktvision.

In Scrum startet das Team die Entwicklung mit unvollständigem Wissen und findet durch das Erstellen von Produktinkrementen heraus, wie die Anforderungen im *Product Backlog* am besten umgesetzt werden können. Entscheidungen werden so bewusst bis zum spätesten verantwortbaren Zeitpunkt (*Last Responsible Moment*) aufgeschoben – dem Zeitpunkt, zu dem eine Entscheidung getroffen werden muss, um den Produkt-erfolg nicht zu gefährden (vgl. [Pop03]). Dadurch gewinnt das Scrum-Team mehr Zeit, um wertvolle Informationen zu sammeln und die bestmögliche Entscheidung zu treffen, anstatt sich frühzeitig auf eine spekulative Lösung festzulegen. Normalerweise sollte die Architekturvision somit nicht festlegen, wie einzelne Anforderungen umgesetzt werden oder welche konkreten Bibliotheken für Spezialfunktionen eingebunden werden.

Vermeiden Sie möglichst zwei häufige Architekturvisionsfehler: *Big Design Up Front (BDUF)* und Planlosigkeit. Eine detaillierte Architekturvision macht es schwer, die Software basierend auf Kunden-Feedback weiterzuentwickeln und das in den Sprints erworbene Wissen in die Ausgestaltung der Architektur einfließen zu lassen. Über keine Architektur zu verfügen, macht es unmöglich, das erste Produktinkrement zielorientiert zu entwickeln. Zusätzlich birgt es die Gefahr hoher architektonischer Refaktorisierungsaufwände. Hierfür gibt es zwei Faustregeln:

- Im Zweifel lieber auf architektonische Festlegungen in der Architekturvision verzichten.
- Architektur- und Technologieentscheidungen möglichst rasch durch Produktinkremente validieren.

die autoren



Stefan Rook

(E-Mail: stefan.roock@it-agile.de)

ist Senior IT-Berater bei it-agile. Seit 1999 arbeitet er in agilen Projekten mit Scrum, XP usw. Er ist Ko-Autor der Bücher „Software entwickeln mit eXtreme Programming“ und „Refactorings in großen Softwareprojekten“.



Roman Pichler

(E-Mail: roman.pichler@romanpichler.com)

ist unabhängiger Berater und international renommierter Scrum-Experte. Er ist Autor der Bücher „Agile Product Management with Scrum“ und „Scrum – Agiles Projektmanagement erfolgreich einsetzen“.

Wünschenswerte Eigenschaften der Vision

Die Architekturvision sollte drei Eigenschaften aufweisen, die wir unter dem Akronym CAS subsumieren (*Clear, Accepted, Short*).

Klar (Clear)

Die Architekturvision sollte die relevanten Entscheidungen so dokumentieren, dass diese für alle Teammitglieder klar verständlich sind. Elemente einer Modellierungssprache wie UML sollten beispielsweise nur dann eingesetzt werden, wenn sie sich eignen und von allen Teammitgliedern verstanden werden. Transparent sollte zudem sein, wie die Architekturvision die Realisierung nicht-funktionaler Anforderungen wie Performanz, Robustheit und Sicherheit unterstützt und wie sich die Entscheidungen auf die Erweiterbarkeit und die Lebenserwartung des Systems auswirken.

Von allen mitgetragen (Accepted)

Die Architekturvision muss gemeinschaftlich akzeptiert sein und von allen Teammitgliedern mitgetragen werden. Nur wenn die Vision ein gemeinsames Bild darstellt, ist eine effektive Teamarbeit möglich. Gilt

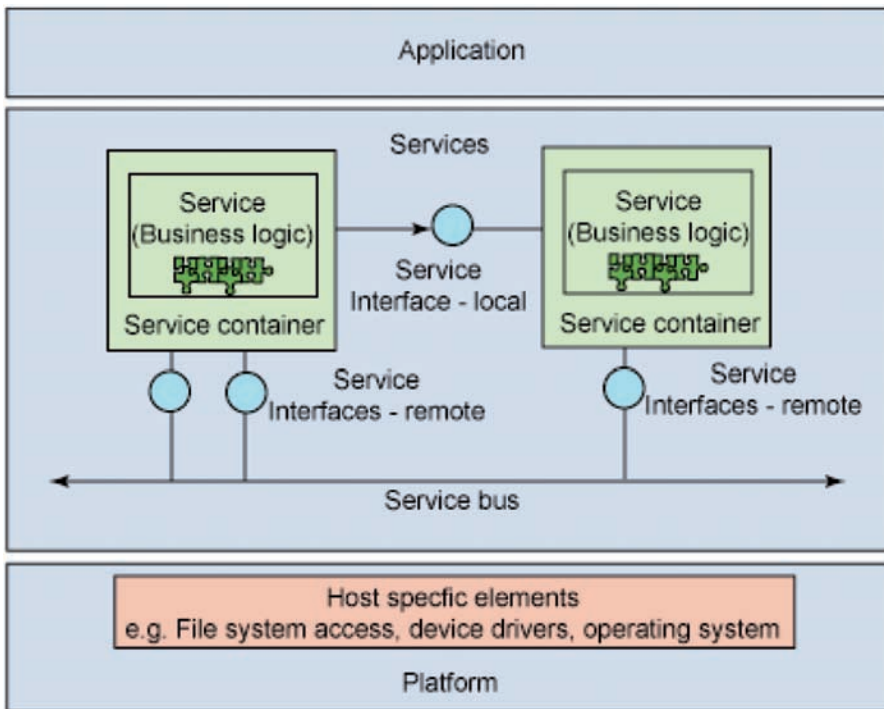


Abb. 1: Beispiel einer Architekturvision.

dies nicht, verfolgen die Teammitglieder ihre eigenen Vorstellungen.

Kurz und bündig (Short)

Weniger ist mehr: Die Architekturvision sollte kein umfangreiches Dokument sein, sondern sich auf wenigen Seiten bzw. auf einem Flipchart oder einem White-Board darstellen lassen. Ziel ist es, die kritischen Entscheidungen festzuhalten – und nicht ein vollständiges Modell zu erstellen. Nicht umsonst stellt das Agile Manifest für Softwareentwicklung fest: „Einfachheit – die Kunst, die Menge nicht getaner Arbeit zu maximieren – ist essenziell“ (vgl. [Bec01]).

Beispiel einer Architekturvision

Betrachten wir als Beispiel die Architekturvision für eine VoIP-Software, an der wir mitgewirkt haben. Die Software hatte das Ziel, Telekommunikationsdienste leicht und durch Fernwartung – auch im laufenden Betrieb – auf dem VoIP-Telefon installieren zu können. Hierzu skizzierte die Vision ein dreischichtiges Software-system mit einer Applikations-, einer Service- und einer Plattformschicht, wie **Abbildung 1** zeigt (vgl. auch [Sid06]).

- Die *Applikationsschicht* stellt die Umgebung für Anwendungen wie

Adressbuch oder Kalender zur Verfügung.

- Die *Serviceschicht* enthält neben den Systemdiensten eine serviceorientierte Komponenten-Middleware. Diese unterstützt unter anderem das Registrieren, das Starten und Stoppen von Diensten sowie die Kommunikation zwischen den Diensten.
- Die *Plattformschicht* abstrahiert vom Betriebssystem und ermöglicht das Deployment der Software auf einem Telefon und auf einem Standard-PC als so genannter Soft Client.

In der Abbildung ist schön zu sehen, dass sich die Architekturvision nur auf die kritischen Aspekte der Software fokussiert. Alle anderen Aspekte bleiben zunächst offen. Die Detaillierung findet während der Entwicklung statt. Einige Teams schreiben die Architekturvision kontinuierlich fort. **Abbildung 2** zeigt die *fortgeschriebene* Architekturvision des oben genannten Teams nach einigen Sprints – eine initiale Architekturvision dieses Umfangs wäre zu spekulativ und unangemessen.

An dem Beispiel ist gut zu erkennen, dass Architekturvisionen meist in einer Variante mit Flipcharts, Karteikarten und Haftnotizen erstellt werden. Das Team muss im Einzelfall prüfen, ob es sich lohnt, das Ergebnis nachträglich in ein elektronisches Tool zu übertragen. In vielen Fällen ist das jedoch nicht notwendig.

Die Erstellung der Architekturvision

Die Architekturvision wird vor Beginn des ersten (Produktinkrement erzeugenden) Sprints unter Verwendung von Prototyping erstellt und somit durch (Wegwerf-) Prototypen gemäß dem Motto „Proof it with code“ validiert (vgl. [Amb02]). Soll Fertigsoftware beispielsweise für die Generierung der Datenbank-Zugriffsschicht eingesetzt werden, so sollte anhand eines Prototypen gezeigt werden, dass die resultierende Software tatsächlich geeignet ist und die entsprechenden Performance-Anforderungen erfüllen kann.

Die Architekturvision sollte die Architektur der jeweils nächsten Produktversion skizzieren. Schließlich gilt in Scrum das agile Mantra: Die einzige Konstante ist



Abb. 2: Eine fortgeschriebene Architekturvision.



Veränderung. Über die nächste Version hinauszuschauen, ist in einem agilen Kontext also kaum möglich.

Die an der Erstellung der Architekturvision beteiligten Mitarbeiter sollten unbedingt auch an der Entwicklung des Produkts als Teammitglieder teilnehmen. So wird das erworbene Wissen effektiv genutzt und Informationsverlusten, Wartezeiten sowie Fehlern vorgebeugt. Außerdem setzt so ein Lernprozess ein: Diejenigen, die die Architekturvision erstellt haben, müssen diese auch umsetzen. Vermeiden Sie daher, dass Experten („Architekten“) die Architekturvision erstellen, die dann nicht Bestandteil des Teams sind. Nehmen Sie sich ein Beispiel an der Entwicklung von Googles „Chrome Browser“: Ben Goodger und Darin Fisher, die beiden Entwickler, die den ersten Browser-Prototypen entwickelt haben, spielten beispielsweise eine wichtige Rolle im Chrome-Entwicklungsprojekt und arbeiteten in den Entwicklungsteams aktiv mit (vgl. [Lev08]). „Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams,“ schreibt das Agile Manifest (vgl. [Bec01]).

Sind mehrere Teams notwendig, um ein Projekt durchzuführen, ändert sich an diesem Ratschlag zunächst nichts. Idealerweise beginnt das Projekt mit einem Team, das die Architekturvision erstellt und das einige Sprints lang an der wichtigsten Funktionalität arbeitet. In der Folge spalten sich von diesem initialen Team weitere Teams ab. Durch den Wechsel von Teammitgliedern des initialen Teams in die abgespaltenen Teams wird das Wissen über die Architekturvision effektiv in alle Teams getragen – so vermeidet man die Gefahr, dass die Projektorganisation die Softwarearchitektur prägt. Letzteres wird auch als „Conway's Law“ bezeichnet (vgl. [Con68]).

Der für die Erstellung der Architekturvision erforderliche Aufwand lässt sich nicht pauschal definieren. Werden wenige neue Funktionen ohne architektonische Auswirkungen in ein existierendes Produkt eingebaut, kann man komplett auf die entsprechenden Architekturvisionsarbeiten verzichten. Wird ein neues, innovatives Produkt entwickelt, kann die Erstellung der Architekturvision mehrere Wochen und in Ausnahmefällen sogar wenige Monate dauern. Als Faustregel gilt: Investieren Sie lieber Tage als Monate, wie die in **Kasten 1** zusammengetragenen Beispiele aus unserer Arbeit belegen.

- Ein Internet-Unternehmen hat in einem Zeitraum von zwei Jahren circa 35 Projekte mit Scrum durchgeführt. Die Projektlaufzeiten betragen zwischen drei und sechs Monaten. In keinem Projekt war es notwendig, mehr als eine Woche in Architekturvisionsarbeiten zu investieren.
- Ein Projekt im Bereich Medizintechnik hatte bereits ergebnislos eine längere Zeit nach klassischem Vorgehen in die Entwicklung eines neuen Produkts investiert, bevor agile Praktiken zum Einsatz kamen. Für die Erstellung der Architekturvision reichte ein zweitägiger Workshop aus.
- In einem Versicherungsprojekt mit einem Umfang von 300 Personentagen wurde die Architekturvision von drei Entwicklern in zwei Tagen erstellt. Sie passte auf zwei Seiten Flipchart, die über den gesamten Projektverlauf im Teamraum hingen.

Kasten 1: Beispiele aus der Praxis.

Die Aufwände für die Erstellung der Architekturvision steigen im Übrigen nicht linear mit der Dauer oder Größe des Projekts. Der sinnvoll zu investierende Aufwand ist dadurch begrenzt, dass wir nur für kurze Zeiträume stabile Aussagen

über die Zukunft machen können – und daran ändert sich durch eine längere Projektlaufzeit nichts.

Die Form der Architekturvision

Für die Architekturvision ist es entscheidend, dass sie vom Team verstanden und mitgetragen wird. Ihre Form ist dabei zweitrangig. Besonders gut für die Darstellung der Architekturvision eignen sich nach unserer Erfahrung White-Boards, Flipcharts und Klebezettel, wie in dem in **Abbildung 3** gezeigtem Beispiel aus einem unserer Projekte. Diese Art der Vergegenständlichung hat mehrere Vorteile:

- Geringer Erstellungs- und Änderungsaufwand sowie geringe Tool-Kosten.
- Nur im persönlichen Gespräch erklärbar, d.h. die Architekturvision eignet sich nicht als Ersatz für die direkte Kommunikation.
- Hohe Sichtbarkeit, da sich die erstellten Flipcharts gut im Teamraum aufhängen lassen: So ist dem Team die Architekturvision jederzeit gegenwärtig und es fällt sehr schnell auf, wenn die aktuelle Entwicklung und die Architekturvision nicht mehr zusammenpassen.
- Vermeidung des Eindrucks einer perfekten und somit starren Architekturvision.

Überlegen Sie daher, ob Sie die Vision wirklich elektronisch vorhalten müssen, und nutzen Sie gegebenenfalls White-Boards,

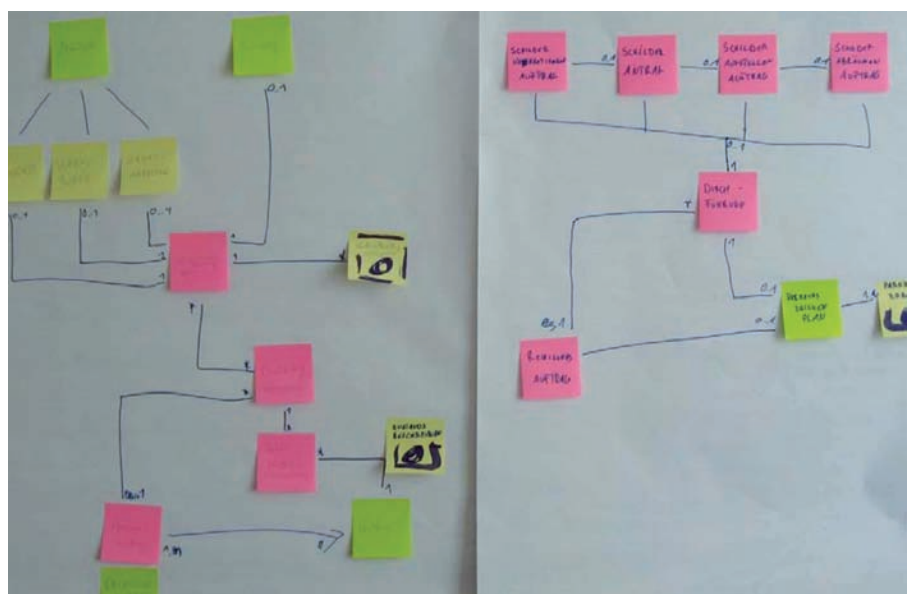


Abb. 3: Architekturvision mit Haftnotizen auf Flipchart.

Flipcharts und Klebezettel zusätzlich zu elektronischen Tools.

Fallen

Vermeiden Sie bei der Erstellung der Architekturvision nach Möglichkeit die folgenden Fallen:

Umbenennen statt umdenken

Gerade bei klassisch geprägten Teams besteht die Gefahr, dass das Konzept der Architekturvision in Richtung Wasserfall missverstanden oder missbraucht wird. Nach wie vor wird die ganze Entwurfsarbeit vor Beginn der Entwicklung durchgeführt, lediglich das Etikett hat sich geändert: Die Analysephase heißt jetzt Architekturvisionserstellung.

Als Faustregel empfehlen wir: Erstellen Sie die Architekturvision in Tagen und nicht in Wochen. Beschränken Sie auch die kritischen Aspekte der Architektur und haben Sie Mut zur Lücke. Wenn Sie für die Erstellung der Architekturvision auch nur annähernd so lange brauchen wie für die klassische Analysephase, machen Sie etwas falsch.

Architekturvision ohne agile Entwicklungspraktiken

Scrum verschiebt die wesentlichen Architekturaufwände in die Entwicklungssprints. Nur ein Bruchteil dessen, was in Wasserfall-Projekten vor Beginn der Entwicklung in die Architektur investiert wird, geschieht vor dem ersten Entwicklungssprint und findet seinen Niederschlag in der Architekturvision. Damit diese Verkürzung der Vorplanung funktioniert, muss das Team die agilen Ent-

wicklungspraktiken – wie kontinuierliche Integration, Testautomatisierung, testgetriebene Entwicklung usw. – beherrschen (vgl. [Pic11], [Roo04]). Werden die Entwicklungspraktiken nicht konsequent eingesetzt, kann auch eine sehr gute Architekturvision eine Explosion der Entwicklungsaufwände nur verzögern, aber nicht aufhalten. Sorgen Sie also dafür, dass das Team die agilen Entwicklungspraktiken beherrscht und bewusst entscheidet, wann es sie einsetzt.

Zusammenfassung

Die Architekturvision ist eine wichtige Voraussetzung für zielgerichtetes und nachhaltiges Entwickeln. Für die Vision gilt: Mut zur Lücke! Beschränken Sie sich auf die Entscheidungen, die unbedingt vorab getroffen werden müssen, damit ein Rahmen für inkrementelles Entwickeln aufgespannt wird. Formulieren Sie die Architekturvision kurz und bündig, denn es gilt: Weniger ist mehr. ■

Literatur & Links

[Amb02] S. Ambler, Agile Modeling. Effective Practices for Extreme Programming and the Unified process, Wiley 2002

[Bec01] K. Beck et al., Manifesto for Agile Software Development, siehe: <http://www.agilemanifesto.org>

[Con68] M.E. Conway, How Do Committees Invent?, in: Datamation, April 1968

[Lev08] S. Levy, Inside Chrome: The Secret Project to Crush IE and Remake the Web, in: Wired Magazine 16.10.2008

[Pic10] R. Pichler, Agile Product Management with Scrum: Creating Products That Customers Love, Addison-Wesley, 2010

[Pic11] R. Pichler, S. Roock (Hrsg.), Agile Entwicklungspraktiken mit Scrum, dpunkt.verlag 2011

[Pop03] M. und T. Poppendieck, Lean Software Development: An Agile Toolkit for Software Development Managers, Addison-Wesley 2003

[Roo04] S. Roock, M. Lippert, Refactorings in großen Softwareprojekten. Komplexe Restrukturierungen erfolgreich durchführen, dpunkt.verlag 2004

[Roo09] S. Roock, Architekturvision und inkrementeller Entwurf, Konferenzvortrag bei „Scrum Gathering Munich“ 2009, siehe:

<http://prezi.com/ptx3bx9lxh4c/architekturvision-und-inkrementeller-entwurf-scrumgathering-munich-2009/>

[Sid06] J. Siddle, VikingPlop 2006, published via IBM developerWorks:

<https://www.ibm.com/developerworks/architecture/library/ar-patstor/>

[Sch04] K. Schwaber, Agile Project Management with Scrum, Microsoft Professional 2004