EXPERIMENTIEREN STATT SPEKULIEREN: ARCHITEKTURMODELLIERUNG UND SPIKES IN AGILEN PROJEKTEN

Agile Projekte versuchen, Anforderungen möglichst schnell in lauffähige Software umzusetzen und dabei regelmäßig potenziell auslieferbare Produktinkremente zu erzeugen. Voraussetzung für eine zielgerichtete Entwicklung ist aber eine klare Vorstellung von der Softwarearchitektur. Wie viel Architekturmodellierung ist notwendig in agilen Projekten? Wann findet diese statt? Wer nimmt sie vor und wie viel Zeit wird benötigt? Der Artikel diskutiert diese Fragen und greift dabei auf die Erfahrungen der Autoren mit agilen Produktneuentwicklungsprojekten im Bereich Enterprise und Embedded zurück.

Funktionierende Software und Architekturmodellierung

Ein wichtiges Ziel agiler Softwareentwicklung ist die Optimierung der Wertschöpfung durch die Vermeidung verschwenderischer und nicht Wert schöpfender Aktivitäten: Deswegen hält das Agile Manifest funktionierende Software für wichtiger als umfangreiche Dokumentation (vgl. [Bec01]). Analyse, Design und Modellierung sind per se keine Tätigkeiten, die Wert generieren: Welcher Kunde bezahlt schon gerne für die Erstellung von UML-Diagrammen am Whiteboard? Agile Methoden wie *Scrum* (vgl. [Bed02]) oder *eXtreme Programming (XP)* (vgl. [Bec04]) versuchen daher, dysfunktionale Analyse- und Designarbeitsweisen abzulegen und möglichst schnell potenziell auslieferbare Produktinkremente zu entwickeln:

- Anforderungen werden nicht länger vollständig erfasst und zu Projektbeginn eingefroren.
- Auf eine extensive Analyse- und Designphase als Voraussetzung für die Kodierung wird verzichtet.
- Das Design wird möglichst schnell in Tests und Code umgesetzt.
- Schwergewichtige Modellierungswerkzeuge, die Spezialwissen erfordern und den Kunden oder Produktverantwortlichen von einer unmittelbaren Steuerung des Projekts ausschließen, werden vermieden.

Nichtsdestotrotz nehmen Design und Modellierung eine prominente Stellung in agilen Projekten ein: Um ein gemeinsames Verständnis des Aufbaus des Softwaresystems und der Interaktion wichtiger Subsysteme/Komponenten herbeizuführen, benötigt man ein Architekturmodell (siehe Abb. 1). Des Weiteren ermöglicht ein Architekturmodell, Design- und Technologieoptionen gemeinsam mit Kunden oder Produktverantwortlichen gezielt abzuwägen und Entscheidungen zur Maximierung der Wertschöpfung zu treffen. Ein Architekturmodell bildet ferner die Grundlage für eine realistische Schätzklausur (*Planning Poker*) und die Erstellung eines ersten möglichst realistischen Softwareentwicklungsplans (*Release Plan*). Zu guter Letzt erlaubt ein Architekturmodell die Auswahl der bestmöglichen Projektorganisation: Sollen sich die Teams an Leistungsmerkmalen (*Features*) oder an Komponenten ausrichten?

Modellierung ist also auch in agilen Projekten wichtig. Zugleich beschränken agile Methoden Design- und Modellierungsaktivitäten auf ein Minimum gemäß dem Motto „So viel wie nötig, so wenig wie möglich". Wie viel Modellierung zu Beginn eines agilen Projekts ist jedoch ausreichend? Wie können wir abschätzen, wie viel Zeit wir für die Erstellung eines Architekturmodells verwenden müssen? Handelt es sich um ein paar Stunden, um einen Tag, um Wochen oder gar Monate?

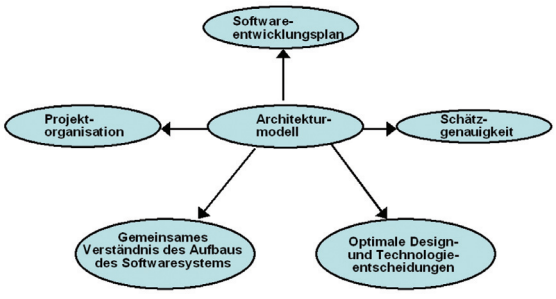

Abb. 1: Die zentrale Stellung des Architekturmodells

Innovation, Risiko und Experimentieren

Der Aufwand für die Erstellung eines Architekturmodells hängt direkt mir dem Innovationsgrad des Projekts zusammen (siehe auch Abb. 2). Neuentwicklungen weisen typischerweise einen höheren Innovationsgrad auf als Weiterentwicklungen oder Wartungsprojekte. Für Neuentwicklungen ist es entscheidend, ob das Team mit der Domäne und den verwendeten Technologien vertraut ist. Haben die Teammitglieder bereits ein ähnliches Produkt entwickelt? Sind sie mit der Programmiersprache und den eingesetzten Tools vertraut? Beispiele für Projekte mit hohem Innovationsgrad sind Entwicklungen von eingebetteten Systemen, von Betriebssystemen oder von spezifischen

die autoren

Roman Pichler (www.romanpichler.com) hilft als Berater, Mentor und Trainer Unternehmen, agile Praktiken erfolgreich einzusetzen. Er ist „Certified ScrumMaster Trainer" und verfügt über reichlich Erfahrung mit agilen Produktneuentwicklungen.

Jim Siddle (www.jamessiddle.net) hat als Softwareentwickler und -architekt eine Vielzahl innovativer Softwaresysteme mitentwickelt. Er arbeitet für IBMs Forschungs- und Entwicklungszentrum Hursley Park in Großbritannien.

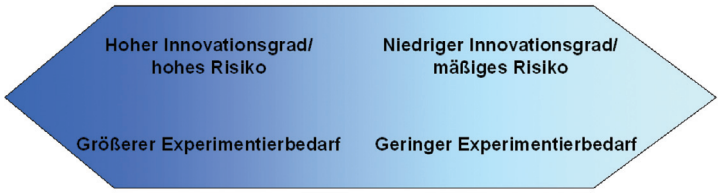

Abb. 2: Innovationskontinuum

Lösungen, beispielsweise virtuelle Maschinen.

Je mehr Innovation in der Softwareentwicklung steckt, desto mehr Risiko nimmt der Kunde/Produktverantwortliche auf sich (vgl. [Pre02]). Die Verwendung neuer, disruptiver Technologien kann einen entscheidenden Wettbewerbsvorteil bieten, kann aber auch zum Scheitern des Projekts führen. Deswegen ist es wichtig, die technologischen Risiken dem Kunden oder Produktverantwortlichen transparent zu machen und ihn in die Architekturauswahl einzubeziehen. Nicht nur der Kunde, auch das Team hat bei einem innovativen Projekt mit Risiken zu kämpfen: Wie können die Teammitglieder sich auf realistische Schätzwerte verständigen, wenn sie nicht wissen, ob Architektur- und Technologieauswahl tragfähig sind?

Die einem Projekt innewohnende Innovation bedingt auch die Art- und Weise der Architekturerstellung:

- Handelt es sich um ein Wartungsprojekt, so ist es möglicherweise ausreichend, das existierende Modell zu erweitern, ohne dies durch die Erstellung eines Prototypen zu validieren.
- Handelt es sich um eine Neuentwicklung, so ist es ratsam, dediziert eine Experimentierphase einzuplanen, in der das Team mit Kunden/Produktverantwortlichen ein gemeinsames Verständnis der wichtigen funktionalen und nicht-funktionalen Anforderungen entwickelt, eine Architekturvision ableitet und diese durch gezieltes Prototyping (in der agilen Entwicklung als *Spikes* bezeichnet) validiert und zu einem Modell weiterentwickelt.

Das Ergebnis der Experimentierphase ist – neben Wegwerftest und -code – ein solides Architekturmodell, das als grober Bauplan zur Erstellung des Softwaresystems und als Grundlage für die Erstellung eines realistischen Entwicklungsplans dient. Interessanterweise kennt XP eine Experimentierphase, auch wenn für deren Notwendigkeit und Dauer wenig Empfehlungen existieren. FDD (vgl. [Pal02]) und Scrum hingegen kennen keine dedizierte Experimentierphase.

Die explizite Beplanung einer Experimentierphase für innovative Softwareentwicklungsprojekte erlaubt dem Kunden/Produktverantwortlichen eine klare Investitionsentscheidung: Wie viel Zeit und Geld ist der Kunde bereit auszugeben und wie viel Risiko ist er bereit, in Kauf zu nehmen? Neben der Risikominimierung dient die Experimentierphase auch der Exploration und Generierung von Ideen, die die Wettbewerbsfähigkeit des späteren Produkts steigern können (vgl. [Tho03]). Google hat beispielsweise das Experimentieren institutionalisiert: Googles Mitarbeiter verwenden bis zu 20% ihrer Zeit für eigene Projekte (*Pet Projects*), die oft maßgeblich zur Produktentstehung beitragen, wie beispielsweise im Fall von „Gmail" (vgl. [Pop06]). Dabei kann es sinnvoll sein, die Experimentierphase mit einem separaten Budget zu planen und möglicherweise als Innovations- oder Vorfeldprojekt zu managen (siehe Abb. 3). Wichtig ist es jedoch, darauf zu achten, dass das Team, das später die Softwareentwicklung vornimmt, auch die Experimentierphase bestreitet. Sonst droht der Verlust wichtigen Wissens und wichtiger Erkenntnisse. Seien Sie sich bewusst, dass der Verzicht auf eine Experimentierphase zwar den Anschein erwecken kann, eine innovative Softwarelösung schneller und kostengünstiger umsetzen zu können. In Wirklichkeit führt ein Verzicht aber häufig zu architektonischen Refaktorisierungsaufwänden, die den Projektfortschritt (*Velocity*) verlangsamen sowie Entwicklungs- und Wartungskosten erhöhen. Schlimmstenfalls kann eine fehlende Experimentierphase das späte Scheitern eines Projekts bedingen.

Agile Methoden unterscheiden sich in ihren Empfehlungen zur Architekturmodellierung: *Scrum* gibt sich agnostisch und rät, im Vorfeld der initialen Schätz- und Planungsklausur genügend Informationen zu sammeln, um die Entwicklung realistisch planen zu können. XP-Teams entwickeln zu Beginn eines Projekts eine Metapher des Softwaresystems. Häufig werden hierzu (Wegwerf-)Prototypen, so genannte *Spikes*, entwickelt, die riskante Bereiche der Architektur explorieren. *Feature-driven Development (FDD)* legt Wert auf die Erstellung eines Modells zu Projektbeginn, das als Roadmap zur Erstellung des Softwaresystems dient. FDD empfiehlt hierbei den Einsatz einer UML-Variante.

Kasten 1: Agile Methoden und Architekturmodellierung

Architekturmodellierung ist Teamarbeit

Die Erstellung eines Architekturmodells sollte vom gesamten Team unter enger Einbeziehung des Kunden vorgenommen werden. So stellen Sie sicher, dass die Ideen aller Teammitglieder berücksichtigt werden und sich alle Mitglieder mit dem Modell identifizieren (*Collective Ownership*). Das Architekturmodell sollte so einfach wie möglich gehalten sein (*Simple Design*). Vermeiden Sie aber simplifizierende Modelle. Designentscheidungen werden in agi-

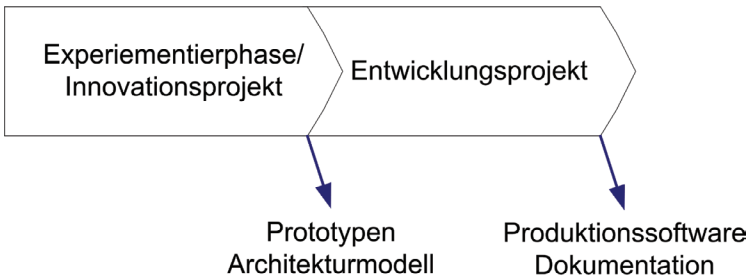

Abb. 3: Innovations- und Entwicklungsprojekt

Eine *Architekturvision* ist eine erste, grobe Idee der Strukturierung des Softwaresystems und der zur Umsetzung benötigten Technologien.
Ein *Architekturmodell* ist ein tragfähiger, grober Bauplan, der typischerweise die Verantwortlichkeiten und Interaktion der wichtigen Softwarebausteine regelt, wichtige *Cross-cutting Concerns* adressiert sowie *Deployment* und Technologieauswahl beschreibt. Das Architekturmodell bildet die Grundlage für die Erstellung eines ersten realistischen Entwicklungsplans. Es weist eine größere Detaillierung als eine Architekturvision auf.

Kasten 2: *Architekturvision versus Architekturmodell*

len Projekten bis zum letzten vernünftigen Zeitpunkt aufgeschoben, um so möglichst viel erworbenes Wissen einbringen und leichter auf Anforderungsänderungen reagieren zu können (vgl. [Pop03]). Diese Praktik wird auch als *Just-in-time Design* bezeichnet (vgl. [Amb02]). Unabhängig davon, welches Medium Sie zur Erfassung des Architekturmodells einsetzen, sollten schnelle Umsetzungen von Änderungen durch alle Teammitglieder möglich sein. Whiteboard und Papierkarten oder Zettel, die Klassen mit ihren Verantwortlichkeiten und Beziehungen ausdrücken (*Class/Responsibilties/Collaborators, CRC*), sind hierfür oft besser geeignet als teure Modellierungssoftware.

Fazit

Architekturmodellierung spielt in agilen Projekten eine wichtige Rolle, insbesondere dann, wenn es sich um Projekte mit hohem Innovationsgrad wie Produktneuentwicklungen handelt. Für die Architekturmodellierung im agilen Kontext gilt *„Prove it with Code"* (vgl. [Amb02]). Eine dedizierte Experimentierphase ist oft wichtig, um die eingesetzten Technologien zu explorieren und Ideen zur Umsetzung der Software zu generieren. Die Planung einer Experimentierphase ist dabei ein Balanceakt: Agile Projekte sollten so wenig wie möglich und so viel wie nötig experimentieren. Konkret kann dies einen Zeitraum von wenigen Tagen bis zu mehreren Monaten bedeuten und muss projektspezifisch unter Berücksichtigung des Innovationsgrads entschieden werden. ■

Literatur & Links

[Amb02] S.W. Ambler, Agile Modeling: Effective Practices for Extreme Programming and the Unified Process, John Wiley & Sons, 2002

[Bec01] K. Beck et al, Manifesto for Agile Software Development, 2001, siehe: www.agilemaifesto.org

[Bec04] K. Beck, C. Andres, Extreme Programming Explained: Embrace Change, Addison-Wesley. 2004

[Bed02] M.A. Beedle, K. Schwaber, Agile Software Development with SCRUM, Prentice Hall, 2002

[Pre02] P.G. Smith, G.M. Merritt, Proactive Risk Management: Controlling Uncertainty in Product Development, Productivity Press, 2002

[Pal02] S. Palmer, A Practical Guide to Feature Driven Development, Prentice Hall, 2002

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

[Pop06] M. und T. Poppendieck, Implementing Lean Software Development: From Concept to Cash, Addison-Wesley, 2006

[Tho03] S.H. Thomke, Experimentation Matters: Unlocking the Potenzial of New Technologies for Innovation, Harvard Business School Press, 2003