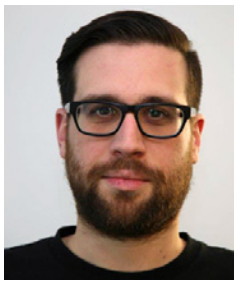
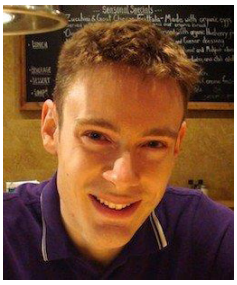

Dominik Rost, Balthasar Weitzel, Matthias Naab und Torsten Lenhart

(matthias.naab@iese.fraunhofer.de)
arbeiten als Berater und Forscher zum Thema Softwarearchitektur am Fraunhofer IESE (Institut für Experimentelles Software Engineering). Sie arbeiten mit Kunden unterschiedlichster Branchen an innovativen Produkten und der Verbesserung von Altsystemen. Neben der direkten Verbesserung der Produkte forschen sie auch auf dem Gebiet der Optimierung von Methoden und treffen dabei häufig auf das Spannungsfeld zwischen Architektur und Agilität.

Durch kleine Bausteine direkter zum Ziel
Architektur-Best-Practices für agile Entwicklung

Agile Entwicklung und Softwarearchitektur sind nicht mehr die Feinde, die sie noch vor einigen Jahren zu sein schienen. Mittlerweile herrscht weitgehende Einigkeit, dass Architekturarbeit in irgendeiner Form auch bei agilen Projekten notwendig ist. Wie aber diese Architekturarbeit aussehen sollte, ist längst nicht so klar. Wir stellen den Bezug zwischen agiler Entwicklung und Architekturarbeit her und präsentieren unsere Kernidee, Best Practices aus klassischen Architekturmethoden zu extrahieren und leichtgewichtig für agile Projekte aufzubereiten. Außerdem präsentieren wir Erfahrungen aus der praktischen Anwendung der Best Practices in agilen Projekten.

Eine Architektur gibt es immer: Fragt sich nur woher

In den letzten Jahren ist agile Entwicklung zur führenden Entwicklungsmethode in vielen Branchen geworden. Davor gab es sehr viele Diskussionen, ob agile Entwicklung und Softwarearchitektur unvereinbare Gegensätze sind. Diese Diskussion scheint endgültig vorbei zu sein, sowohl bei Praktikern als auch in der Forschung. Trotzdem ist immer noch nicht klar, wie Architekturarbeit in agilen Projekten genau aussehen sollte. An diesem Punkt setzen wir an und stellen einen leichtgewichtigen Ansatz vor, mit dem passgenau Architekturarbeit in agilen Projekten eingesetzt werden kann.

Jedes Softwaresystem hat eine Architektur, egal wie es entwickelt wurde. Das heißt, wir müssen uns darum kümmern, wie diese Architektur entsteht und wie Architekturarbeit dazu beiträgt. Dazu charakterisieren wir Architekturarbeit durch drei Bereiche:

- Architekturaktivitäten: Was wird gemacht rund um Architektur?
- Architekturdokumentation: Wie wird Architektur festgehalten?
- Architekturverantwortlichkeiten: Wer macht was rund um Architektur?

Agile Entwicklungsmethoden sind oft viel strikter, als sie auf den ersten Blick erscheinen. Es wird ein strenger iterativer Ablauf mit klaren Regeln vorgegeben, um regelmäßig laufende Software zu liefern. Dabei bringen weder Methoden wie Scrum, die eher auf Projektmanagement fokussieren, noch solche wie XP, die eher auf Implementierung fokussieren, die explizite Arbeit an der Architektur ins Spiel. Trotzdem sind viele agile Projekte sehr erfolgreich.

Bei genauerem Hinschauen zeigt sich oft, dass diese Projekte viel gemeinsam haben: Erfahrene und sehr gute Entwickler in einem eher kleinen Team arbeiten an einer nicht so großen Software in der initialen Entwicklung. Dabei treten einige Komplexitätsfaktoren, die durch Architekturarbeit normalerweise adressiert werden, gar nicht auf, und andere werden einfach durch die Erfahrung der Entwickler gelöst.

Abb. 1: Drei Aspekte von Architekturarbeit

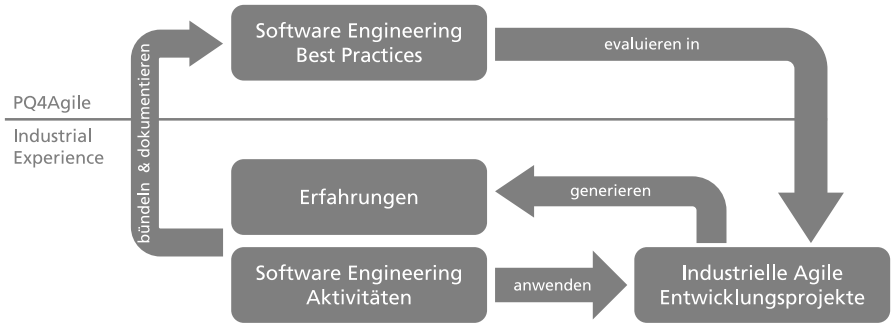

Abb. 2: Entwicklung von Architektur-Best-Practices

Immer häufiger wird agile Entwicklung aber auch in Projekten angewendet, die nicht die beschriebenen Eigenschaften haben. Dann zeigt sich oft schnell, dass die gewünschte Qualität nicht erreicht wird. Spätestens dann wird explizite Architekturarbeit notwendig. An diesem Punkt setzen wir an und stellen Best Practices bereit, mit denen agile Teams ihre Architekturarbeit durchführen können.

Gestaltungsraum für Architekturarbeit in agilen Projekten

Aus unserer Projekterfahrung haben wir drei Bereiche identifiziert (siehe Abbildung 1), die helfen, Architekturarbeit in agilen Projekten einzuteilen und das Verhältnis von Architekturarbeit und agiler Entwicklung weiter zu charakterisieren.

Ein Bereich sind konkrete **Architekturaktivitäten**, darunter fassen wir nicht nur das Treffen und Dokumentieren von Architekturentscheidungen zusammen, sondern beispielsweise auch die Analyse von Architekturanforderungen, Evaluieren von Lösungen, Unterstützung bei der Umsetzung sowie die Überprüfung der Architekturtreue.

Dabei ist der Zeitpunkt der Durchführung der Aktivität geeignet, um diese weiter einzuordnen. Ein möglicher sehr früher Zeitpunkt wäre noch vor der eigentlichen Entwicklung als eine vorgelagerte Planungsphase, alternativ dazu bietet sich ein Sprint 0 an. Während der Iteration können Architekturaktivitäten entweder in Spikes stattfinden, während der Planung für die nächste Iteration oder sehr kurzfristig während der Umsetzung. Der Fokus der durchgeführten Aktivitäten können beispielsweise bestimmte kritische Aspekte des Systems, ausgewählte User-Stories und Epics oder das komplette Produkt sein.

Ein weiterer Bereich ist die Art der **Architekturdokumentation**, also die Manifestierung der Ergebnisse von Architekturaktivitäten in Form von Entscheidungen, Sichten und Dokumentation für die Entwicklung. Das Spektrum reicht hier von reiner Erinnerung über Whiteboard Zeichnungen zu Architekturdokumenten und formalen Architekturmodellen. Hierbei ist in der Praxis natürlich eine Mischung all dieser Ansätze am häufigsten anzutreffen, beispielsweise werden Whiteboard-Sketches durch Architekturdokumente ergänzt, die wichtige Architekturentscheidungen beschreiben.

Der letzte Bereich zur Einteilung von Architekturarbeit in agilen Projekten ist die Verteilung von **Architekturverantwortlichkeiten**, also die Ausführung von Architekturaktivitäten durch bestimmte Rollen und Personen. Dabei ist es besonders im agilen Umfeld verbreitet, diese Verantwortlichkeit beim kompletten Team zu sehen. Eine Alternative dazu ist eine Gruppe von Architekten im Team, oder auch eine einzelne Person, die diese Aufgaben übernimmt. Gerade auch in größeren Unternehmen ist ein separates Architekturteam, das dem Entwicklungsteam zuarbeitet, nicht ungewöhnlich.

Höhere Qualität durch Architekturbausteine in agiler Softwareentwicklung

In Projekten mit Industriekunden haben wir die Erfahrung gemacht, dass wir mit einer Integration von Softwarearchitektur-Praktiken und agiler Softwareentwicklung Qualitätsattribute schneller, einfacher und vor allem systematischer erreichen können. Um die Praktiken, die wir erfolgreich angewandt haben, aufzubereiten, zu bündeln und Entwicklern in agilen Projekten zur Verfügung zu stellen, haben wir das Projekt **PQ4Agile** (Produkt-Qualität für Agile Softwareentwicklung) ins Leben gerufen. Damit wollen wir außerdem eine Basis für die Diskussion mit der Community und einen geeigneten Rahmen schaffen, um Praktiken in dieser Weise aufbereitet im Praxiseinsatz zu testen (siehe Abbildung 2).

Um dies zu ermöglichen, greifen wir in PQ4Agile auf etablierte Softwareengineering-Methoden zurück. Viele dieser Methoden sind vielfach erprobt und werden kontinuierlich weiterentwickelt. Unsere Kernidee ist es, solche zusammenhängenden Methoden in kleine Bausteine, also Best Practices, zu zerlegen, die agile Softwareentwickler in Projekten einfach aussuchen, ausprobieren und einsetzen können, und das ohne dafür gleich einen anderen Entwicklungsprozess adaptieren zu müssen.

Da diese Praktiken nicht notwendigerweise kompatibel zu agilen Entwicklungsvorgehen sind, müssen wir sie auf die Besonderheiten von agilen Prozessen, wie beispielsweise kurze Iterationszyklen oder enge Interaktion mit Stakeholdern, anpassen.

Stefan Toth hat mit seinem Buch „Vorgehensmuster für Softwarearchitektur“ [Toth14] einen ähnlichen Weg eingeschlagen. Wir bauen darauf ein Kompendium auf, aus dem Softwareentwickler geeignete Best Practices für ihre aktuelle Situation auswählen, aber auch vorhandene diskutieren und neue Best Practices einstellen können. Es entsteht unter http://wiki.pq4agile.de, und wir laden die Community explizit ein, sich an der Weiterentwicklung zu beteiligen.

Bisher wurden 5 Architektur-Best-Practices beschrieben, darüber hinaus 35 weitere in den Bereichen Requirements Engineering, User Experience sowie Testen und unterstützende Praktiken. Ein Beispiel für die Dokumentation einer Best Practice ist in Abbildung 3 dargestellt. Tabelle 1 zeigt als Beispiel einen Auszug der Best Practice „Architekturlösungen im Team entwickeln“.

Unsere Erfahrungen

Die im Rahmen von PQ4Agile erarbeiteten Best Practices wurden von uns bereits in vielen agilen Entwicklungsprojekten eingesetzt und in diesem Zuge immer weiter verfeinert. Die dabei gewonnenen Erfahrungen (siehe auch [KrNaHu14]) lassen sich wiederum in die bereits oben genannten Bereiche Architekturaktivitäten, Architekturdokumentation und Architekturverantwortlichkeiten kategorisieren.

In Bezug auf **Architekturaktivitäten** ist es zunächst wichtig, mit einem weit ver-

Kontinuierliche Architekturbewertung durchführen	Architekturaktivitäten
Ziele: (1) Inkonsistenzen und offene Punkte in der Lösung identifizieren und deren frühzeitige Bearbeitung ermöglichen, (2) Architekturlösungen nachhaltig bewahren, (3) Konfidenz in die Eignung der entwickelten Architekturlösungen erreichen, (4) Wissensverteilung über Architekturkonzepte im Team fördern	
Motivation: In fast jedem Entwicklungsprojekt sind die gegenseitigen Auswirkungen und Abhängigkeiten von Lösungskonzepten oft nur schwierig vollständig zu überblicken. Wird ein Lösungskonzept für neue oder geänderte Anforderungen entwickelt, kann es daher einerseits dazu kommen, dass Lösungskonzepte als geeignet eingestuft werden, dabei aber wesentliche Punkte übersehen werden, die die Erfüllung der bearbeiteten Anforderungen unmöglich machen. Andererseits kann ein grundsätzlich geeignetes Konzept solch große Auswirkungen auf andere Lösungskonzepte haben, dass die Erfüllung von wichtigen, querschnittlichen Anforderungen beeinträchtigt wird. Inkonsistenzen und ungeeignete Konzepte werden dann von Entwicklern häufig zufällig während der Implementierung aufgedeckt oder es kommt zu unerklärlichem Verhalten oder Fehlern. Das Ergebnis ist ein weniger wartbares System und zusätzlicher, ungeplanter Aufwand für Korrekturarbeiten. Architekturbewertungen können helfen, solche Inkonsistenzen in Architekturlösungen früher aufzudecken und mit den Auswirkungen umzugehen. Eine stark fokussierte Architekturbewertung für die wichtigsten Anforderungen kann auch mit wenig Zeitaufwand gut durchgeführt werden. Werden solche Bewertungen wiederholt und regelmäßig durchgeführt, stellen sie einen großen Mehrwert für agile Projekte dar.	
Input: Eine Menge von Anforderungen, für die das Konzept entwickelt wurde. Zusätzliche wichtige oder querschnittliche Anforderungen	*Output:* Eine Einschätzung über die Eignung des Systems hinsichtlich der analysierten Anforderungen. Eine Menge von offenen Punkten und Verbesserungspotenzialen für das System
Beschreibung: Lösungskonzepte werden für eine Menge von Anforderungen häufig während eines Sprints in Vorbereitung für die Umsetzung im Folgesprint erarbeitet. Deshalb sollte man sich klar machen, gegen welche Anforderungen man bewerten will. Das sind natürlich die, für die das Konzept entwickelt wurde, und frühere, die dadurch nicht beeinträchtigt werden dürfen. Während der Bewertung übernimmt ein Entwickler die Rolle des Assessors. Idealerweise hat dieser nicht direkt bei der Entwicklung und Realisierung des Lösungskonzeptes für die untersuchten Anforderungen mitgewirkt, um nicht voreingenommen zu sein. Einer oder mehrere Entwickler sind die Befragten. Die Anforderungen sollten so konkret vorliegen, dass sie auch geprüft und diskutiert werden können. Für die Bewertung setzt sich der Assessor mit den Befragten zusammen. Diese erklären dem Assessor für jede ausgewählte Anforderung, wie die Lösung die jeweiligen Anforderungen erfüllt. Der Assessor fordert das Team heraus und versucht, vergessene Aspekte, Inkonsistenzen und offene Punkte zu finden. Für identifizierte offene Punkte oder Risiken werden Vorschläge für Stories formuliert, die mit dem Product Owner und dem Team besprochen und priorisiert werden. Eine Kurzfassung der Bewertungsergebnisse kann den Teammitgliedern beim folgenden Stand-up (oder ähnlichem) Meeting mitgeteilt werden. Die ausführlichen Ergebnisse können beispielsweise während des nächsten Planungsmeetings (erster Teil) am Anfang der folgenden Iteration präsentiert werden, um etwaige Nacharbeiten zu koordinieren. Für den Erfolg dieser Best Practice ist es essenziell, die richtigen Teammitglieder für die teilnehmenden Rollen auszuwählen, insbesondere für den Assessor, der einen sehr guten technischen Gesamtüberblick haben muss. Die Befragten müssen bereit sein, offen über die Lösungen zu sprechen, insbesondere wenn Verbesserungspotenzial diskutiert wird. Für die Bearbeitung identifizierter Verbesserungspotenziale sollten explizit Stories erstellt und eingeplant werden. Die Zeit zur Vorbereitung und Durchführung der Bewertung muss als Teil der Entwicklung angesehen werden und damit als wertvolle Investition in die Qualität des Produktes. Ansonsten besteht die Gefahr, dass die Best Practice nicht sinnvoll durchgeführt wird und damit kein Mehrwert für das Projekt erzielt wird.	

Tabelle 1: Best-Practice-Beispiel „Architekturlösungen im Team entwickeln“

breiteten Irrglauben aufzuräumen: Agile Entwicklung impliziert keineswegs, dass sämtliche Planungsaktivitäten zugunsten der Umsetzung konkreter Funktionen aufgegeben werden müssen. Tatsächlich wird man im agilen Umfeld nicht gezwungen, immer gleich Kopf voraus ins Wasser zu springen – es wird lediglich darauf Wert gelegt, dass man sich nicht zu sehr mit Trockenübungen aufhält. Konzepte wie der Sprint 0 in Scrum sollten daher genutzt werden, um die architektonische Basis für das zu entwickelnde Produkt zu definieren, ohne dass die Fokussierung auf den Business Value und die generelle Of-

Abb. 3: Best-Practice-Beispiel

fenheit für Änderungen beeinträchtigt werden.

Des Weiteren bietet es sich an, die oftmals in agilen Setups ohnehin etablierten Code Reviews dahin gehend zu erweitern, dass auch die Konformität zu den gemeinsam im Team getroffenen Design-Entscheidungen explizit überprüft wird. Beachten muss man bei der Einführung von Architekturaktivitäten außerdem, dass die häufig ohnehin schon relativ hohe Meeting-Dichte nicht weiter aufgebläht wird, da das die Akzeptanz der neuen Methoden negativ beeinflussen könnte. Stattdessen sollten wenn möglich bereits etablierte Meetings verwendet werden.

Auch hinsichtlich der **Verantwortlichkeiten** für die Architektur konnten wir wertvolle Erfahrungen sammeln: Auf der einen Seite ist zwar klar, dass das Team für die Definition der Architektur und daher auch für die Einführung der von uns beschriebenen leichtgewichtigen Methoden zuständig ist, andererseits muss aber auch bedacht werden, dass die Priorisierung der in einem Zeitintervall umzusetzenden Tasks in der Regel den Stakeholdern beziehungsweise deren Repräsentant (z. B. *Product Owner* in Scrum) obliegt. Hier kann ein Konflikt entstehen, da für die Etablierung der Best Practices im Normalfall zunächst ein gewisser Anteil der Gesamtkapazität investiert werden muss, welcher entsprechend nicht für die Umsetzung konkreter Features zur Verfügung steht. Aus diesem Grund ist es unabdingbar, die für die Priorisierung Verantwortlichen in den Entscheidungsprozess, welche Methoden eingeführt werden, zu integrieren und sie vom Nutzen eines solchen Investments zu überzeugen.

Aber auch das Team selbst muss gegebenenfalls stärker Verantwortung übernehmen: So werden zum Beispiel Qualitätsattribute in agilen Projekten oft nicht ausdrücklich definiert und es wird stattdessen erwartet, dass diese aus dem jeweiligen Kontext abgeleitet werden können und allen Teammitgliedern mehr oder weniger implizit bewusst sind. Unsere Erfahrung hat gezeigt, dass diese Qualitätsattribute explizit gemacht werden müssen, zum Beispiel indem sie, abhängig von ihrer Art und Wichtigkeit, entweder in die Akzeptanzkriterien einzelner Stories, in die *Definition of Done* oder als spezifische Backlog Items aufgenommen werden. Für die korrekte Umsetzung stehen dann jeweils die an einem solchen Item arbeitenden Team-Mitglieder in der Pflicht.

Ein letzter Punkt in Bezug auf Architekturverantwortlichkeiten betrifft Setups, in denen mehrere agile Teams an einem Produkt zusammenarbeiten. Dort ist die Abstimmung von Architekturentscheidungen über Teamgrenzen hinweg unbedingt zu etablieren, da ansonsten Silolösungen entstehen und die Architektur schon während der Entstehung durch inkompatible Entscheidungen degeneriert – mit unvorhersehbaren Auswirkungen auf die Qualität des Systems. Ansätze, die hier hilfreich sein können, sind zum Beispiel Scrum of Scrums [Suth01] oder das Scaled Agile Framework [Leff07].

Unsere Erfahrungen bezüglich **Architekturdokumentationen** lassen sich folgendermaßen zusammenfassen: Auch hier gilt es zunächst, der verbreiteten Ansicht entgegen zu treten, dass im Kontext agiler Entwicklung generell auf Dokumentation verzichtet werden kann. Oft wird als Beleg für diese These das *Agile Manifesto* und insbesondere der Satz „[Wir schätzen] funktionierende Software mehr als umfassende Dokumentation“ ins Feld geführt. Dabei wird verkannt, dass durch dieses Statement lediglich die Prioritäten verdeutlicht werden sollen, es sich aber keineswegs um eine binäre Entscheidung für das eine und gegen das andere handelt. Soll ein mittel- und langfristig erfolgreiches Produkt entwickelt werden, ist eine prägnante und leichtgewichtige Dokumentation nicht nur erstrebenswert, sondern unerlässlich.

Um die Bereitschaft zur Mitarbeit an der kontinuierlichen Erstellung und Pflege der Dokumentation zu stärken, ist es wichtig, die Hürden hierfür so niedrig wie möglich zu halten. In diesem Zusammenhang hat es sich als äußerst hilfreich erwiesen, für Dokumentationszwecke ausschließlich auf Standard-Tools zurückzugreifen, die ohnehin in der täglichen Arbeit zum Einsatz kommen. Insbesondere die häufig verwendeten virtuellen Agile Boards können hier eine wichtige Rolle spielen, indem beispielsweise Zusammenfassungen von Diskussionen direkt als Kommentar zum jeweiligen Backlog Item hinzugefügt oder separat erstellte Dokumente als Attachements angehängt werden. Weiterhin kann es hilfreich sein, die *Definition of Done* um einen Punkt zu erweitern, der einen angemessenen Dokumentationsstand einfordert.

Fazit

Agile Entwicklung und Architekturmethoden schließen sich keineswegs gegenseitig aus, sondern können vielmehr kombiniert eingesetzt werden, um qualitativ hochwertige Software zeit- und budgetgetreu zu entwickeln. Im Rahmen von PQ4Agile haben wir verschiedene etablierte Softwareengineering-Methoden in die agile Welt portiert und erfolgreich in

der Praxis eingesetzt. Gleichzeitig ist uns bewusst, dass es sich dabei lediglich um erste Schritte handelt und auf diesem Gebiet noch immer ein großes ungenutztes Potenzial schlummert. Mit unserer Arbeit wollen wir daher auch eine gemeinsame Basis schaffen und Forschung und Praxis zur verstärkten Diskussion und Zusammenarbeit motivieren. ■

Literatur & Links

[KrNaHu14] K. Krogmann, M. Naab, O. Hummel, Agile Anti-Patterns. Warum viele Organisationen weniger agil sind, als sie denken, in: Business Technology Magazin, 02/2014

[Leff07] D. Leffingwell, Scaling Software Agility: Best Practices for Large Enterprises, Addison-Wesley Professional, 2007

[Suth01] J. Sutherland, Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies, in: Cutter IT Journal, Vol. 14, No. 12., siehe http://www.controlchaos.com/storage/scrum-articles/Sutherland%20200111%20proof.pdf

[Toth14] St. Toth, Vorgehensmuster für Softwarearchitektur, Hanser Fachbuch, 2014