Dr. Dirk Muthig

(dirk.muthig@lhsystems.com)
leitet das Architecture and Software Lifecycle Management-Team im Profit Center Airline Solutions der Lufthansa Systems AG. Seine Arbeit dreht sich rund um die Definition und Realisierung von geeigneten Systemarchitekturen, Standards und Richtlinien für Softwareprodukte, die von vielen Airlines weltweit genutzt werden.

Agilität – eine Frage der Organisation

Agilität ist im Kern eine Eigenschaft, die an einer Schnittstelle entsteht: die gegenüberliegende Seite wird als agil empfunden oder eben nicht. Agilität bedeutet, dass sie mit überraschender Leichtigkeit und Geschwindigkeit auf Änderungen reagiert. Der typische Fall in der Literatur ist die Schnittstelle zwischen Kunden und einem Entwicklungsteam. Der Kunde hat die Möglichkeit, das weitere Vorgehen kontinuierlich zu beeinflussen – aufgrund der Bewertung des bisher Erreichten oder in Form von Änderungen an Anforderungen, Prioritäten oder sonstigen Kontextfaktoren. Kontinuierliche Beeinflussung ist natürlich nur eine Illusion, die durch eine möglichst schnelle Abfolge von diskreten Zeittakten erreicht wird. Zu Beginn eines Taktes können die Vorgaben für den oder die kommenden Takte (neu) formuliert werden. Agile Methoden beschreiben nun, wie man die Arbeit organisieren kann, um einen sinnvollen Arbeitsrhythmus zu implementieren. Eine Organisation, die es erlaubt Änderungen zwischen den Takten auch direkt zu folgen mit möglichst minimalen Reibungsverlusten. Um die grundsätzlichen Verbesserungen von agilen Methoden herauszuarbeiten, werden sie gerne mit wasserfallartigen Projekten verglichen, die Änderungen nur nach ausgiebigen Vertragsverhandlungen in Ausnahmefällen erlauben. Schnell neigt man dazu, alleine dem Einsatz von agilen Methoden den positiven Unterschied an der Schnittstelle gutzuschreiben. Daraus resultiert, dass eine detaillierte Analyse der Gesamtorganisation zu spät stattfindet. Die häufig weitreichenden, notwendigen Änderungen in einer Organisation werden dann erst in den laufenden Arbeiten provisorisch nachgezogen. Dieser Artikel untersucht die Auswirkungen auf eine Organisation, die hinter einer Schnittstelle steht, an der Agilität gefordert ist. Der Artikel startet zunächst mit einer Betrachtung des klassischen Falls, in dem ein Entwicklungsteam direkt und ausschließlich mit seinem Kunden interagiert. Danach kommen weitere Kunden hinzu, sodass es sich um keine Individualentwicklung mehr handelt. Aufgrund der Größe des Systems wird anschließend eine komplexere Entwicklungsorganisation angenommen, in der mehrere Teams mit unterschiedlichen Aufgaben zu dem System beitragen. Hier schließt sich eine Betrachtung der erreichbaren Agilität an der Kundenschnittstelle an, die von der Agilität, die an den direkten, internen Schnittstellen erreicht wird, abhängt. Danach erweitern wir den Scope und stellen uns mehrere Systeme vor, die zum Teil gleiche Teilsysteme nutzen und bringen damit beide Aspekte, nämlich mehrere Kunden und komplexere Entwicklungsorganisation, zusammen.

Individualentwicklung im Kleinen: ein Team, ein Kunde

Beginnen wir mit dem einfachsten Fall: ein Kunde arbeitet mit einem Entwicklungsteam zusammen, welches alle notwenigen Aktivitäten selbst durchführt (siehe Abbildung 1). Das Projekt wird durch drei Dinge charakterisiert: Anforderungen an das System, Termin an dem das System eingesetzt werden soll und das vorhandene Budget. Änderungen des Kunden können nun auch diese drei Parameter betreffen. Agilität bedeutet nun, auf Änderungen aller Art zu reagieren bzw. grundsätzlich reagieren zu können. Dazu gehört, dass man die Auswirkungen auf die anderen, abhängigen Parameter schnell ableiten und plausibel darlegen kann.

Agilität zielt in erster Linie darauf ab, Änderungen der Anforderungen zu akzeptieren ohne Budget und Termin in Frage zu stellen. Um Verschwendung zu vermeiden, limitiert man „Work in Progress“, fängt also nur ein paar Dinge an, die man erst vollständig abschließt, bevor man weitere Anforderungen angeht. Mit diesem Ansatz kann man alle noch nicht angefangenen Anforderungen ohne weiteres ändern ohne vorherige Arbeiten in Frage zu stellen.

An dieser Stelle muss erwähnt werden, dass man zu Beginn eine tragfähige Architektur definieren muss, die das Grundgerüst darstellt, in dessen Rahmen man dann Anforderungen umsetzt. Demzufolge sind bei Anforderungen in erster Linie funktionale Aspekte gemeint, wenn Änderungen ohne rückwirkende Auswirkungen bleiben sollen.

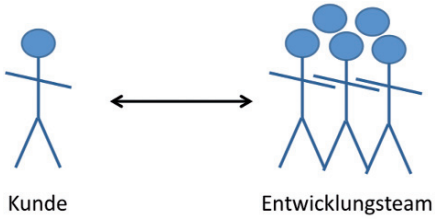

Abb. 1: Individualentwicklung

Reibungsverluste stecken bei diesem Ansatz im Budget. Steckt der Kunde im Vorfeld das gewünschte System eng ab und kalkuliert mit „spitzem Bleistift“, dann ist kaum Raum übrig, in dem man umverteilen kann. Benötigt dann ein Teil mehr Aufwand als gedacht, fehlt es direkt an anderen Ecken. Demzufolge muss der Kunde hier anders herangehen, als über den klassischen Weg des Einkaufs, wo man ein System ausschreibt und den besten Bieter (mit starkem Blick auf die Kostenseite) auswählt.

Grundsätzlich kann man hier höhere Kosten erwarten, diese häufig aber auch über bessere Systeme und Qualität rechtfertigen. Die Wirtschaftlichkeit muss dabei klar aufgezeigt werden, wobei die Fragen bleiben, ob es gelingt den Mehrwert deutlich und überzeugend genug herauszuarbeiten und ob sich die Kundenorganisation diesen Mehrwert überhaupt leisten will.

Dies wird in der Regel nicht bei allen Vorhaben gelingen, zumal das Risiko auch deutlich mehr beim Kunden verbleibt – im Vergleich zu einem Pauschalangebot auf Basis eines Festpreises. Es ist die Verantwortung des Kunden, die aktiven Parts beim agilen Vorgehen auch zu leben: sich kompetent und kontinuierlich mit dem Projekt auch inhaltlich auseinanderzusetzen. Gelingt dies nicht, kann es auch mit agilen Methoden sehr leicht zu Fehlentwicklungen kommen.

Die Erfahrung zeigt, dass ein agiler Ansatz sehr gut funktioniert, wenn bestimmte Voraussetzungen erfüllt sind. Dem Kunden ist die Unsicherheit, wo er überhaupt landen möchte, sehr bewusst. Das eigentlich angestrebte System ist auch deutlich größer, als was man im ersten Wurf erwartet, wodurch ausreichende Manövriermasse zur Verfügung steht, die Agilität erlaubt.

In solch einem Szenario können beide Seiten in das Thema hineinfinden und lernen, Vertrauen aufzubauen und eine langfristig, sehr gute Kooperation zu etablieren, die über die aufbauende Entwicklungsphase hinaus auch darauffolgende Wartungsphasen sehr gut abdeckt. Lediglich die Kostenbetrachtung kann dann Gründe liefern, um irgendwann doch noch zu einem anderen Modell überzugehen.

Produkt aus der Manufaktur: ein Team, möglichst viele Kunden

Ein anderes Modell kann sein, die Individualentwicklung in ein Produkt zu überführen, welches man für viele Kunden pflegt und weiterentwickelt (s. Abbildung 2). Die Erwartung des Kunden ist natürlich, dass man zum einen viele Kostenpunkte über alle Kunden verteilen kann, auf der anderen Seite die grundsätzliche Zusammenarbeit jedoch nicht ändert.

Es ist also Agilität an jeder individuellen Kundenschnittstelle gefordert, die durch ein Entwicklungsteam erbracht wird. Kanban-Systeme berücksichtigen bereits mehrere Quellen von Anforderungen. Jeder Kunde akzeptiert in gewissem Rahmen Abstriche hinsichtlich der Liefertermine seiner Anforderung, um Kostenvorteile zu erzielen. In diesem Fall ist es Aufgabe der Entwicklungsorganisation, die Vorteile stets explizit vor Augen zu führen, um andere Einschränkungen besser motivieren zu können.

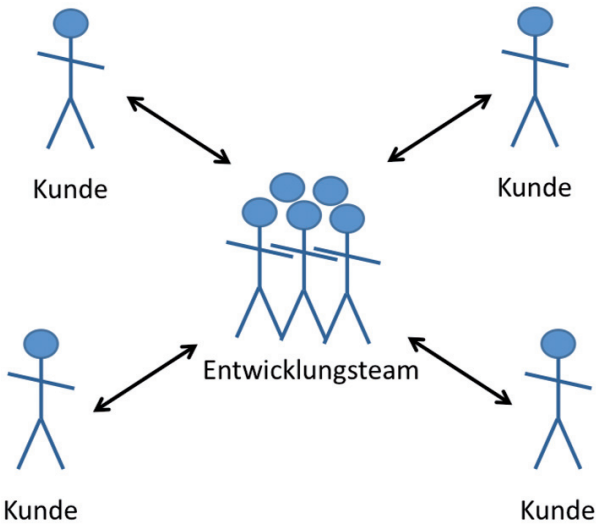

Abb. 2: Produkt aus der Manufaktur

Grundsätzlich wird den verschiedenen Kunden jeweils eine bestimmte Kapazität zugeordnet, die sich aus den einzelnen Budgetanteilen ergibt. An dieser Stelle werden auch die Reibungsverluste sichtbar: bucht ein Kunde eine bestimmte Kapazität, die er dann je nach Bedarf mit Inhalten füttern muss, kann er agiler weiterarbeiten, als wenn er nur bei konkretem Bedarf einen Change Request in Auftrag gibt und diesen dann vom Entwicklungsteam abarbeiten lässt.

Wie bei einer Individualentwicklung ist Agilität nur zu erreichen, wenn man eine gewisse Menge an Überkapazitäten hat, die das Produkt generell weiterentwickeln, wenn es keine konkreten Anfragen der Kunden gibt. Dies müsste man durch höhere Wartungspauschalen oder Ähnliches finanzieren. Zielt man jedoch darauf ab, die laufenden Kosten so gering wie möglich zu halten, wird man in der Regel bei einer Vollauslastung der Entwicklungsmannschaft durch direkt von Kunden beauftragte Änderungen herauskommen.

Da man in der Regel nicht weiß, wie viele Änderungen verteilt auflaufen werden, wird man mit dem Ziel der Kostenminimierung bei einem Team landen, welches klein genug ist, um Volllast sicher zu erreichen. Daraus folgt direkt, dass einige geforderte Änderungen nicht mehr zeitnah realisiert werden können.

Da Kunden nur noch beauftragen, was sie unbedingt brauchen, geht die notwendige Manövriermasse verloren. Zufriedenstellende Agilität ist unter diesen Bedingungen nicht mehr zu erreichen, da jede Änderung eine andere benötigte Änderung verzögert.

Auch in diesem Kontext ist Agilität nur durch Mehrkosten zu erreichen, die man in einigen Fällen durch Produktverbesserungen rechtfertigen kann. Jedoch muss auch hier erst allen Kunden der Mehrwert verdeutlicht werden. Ansonsten muss man nach Wegen suchen, die das dynamische Anpassen der Teamgröße je nach Bedarf erlauben. In ein gut eingespieltes Team neue Mitarbeiter richtig zu integrieren, benötigt jedoch einen signifikanten Vorlauf und Aufwand; ist dies einmal erfolgreich gelungen, will man dies sicherlich nicht gleich durch einen Abbau bei Bedarfsrückgang wieder zunichtemachen.

Generell sind die Möglichkeiten auf diesem Weg dynamisch zu skalieren relativ begrenzt. Neben der Trägheit durch notwendigen Einarbeitungsaufwand, gibt es generell eine maximale Größe, die noch eine sinnvolle Zusammenarbeit im Team erlaubt. Demzufolge muss man nach anderen Möglichkeiten suchen: geschickte Kategorisierung der Aufgaben in mehrere Arbeitsfelder oder Aufteilung zwischen mehreren Teams.

Arbeitsteilung: mehrere Arbeitsfelder

Eine Skalierung der Entwicklungskapazität kann nur erfolgen, wenn man die notwendigen Arbeiten in kleinere, wohldefinierte Arbeitsschritte unterteilt und so eine Basis für eine systematische Arbeitsteilung schafft (s. Abbildung 3). Für alle Schritte muss man die notwendigen Fähigkeiten

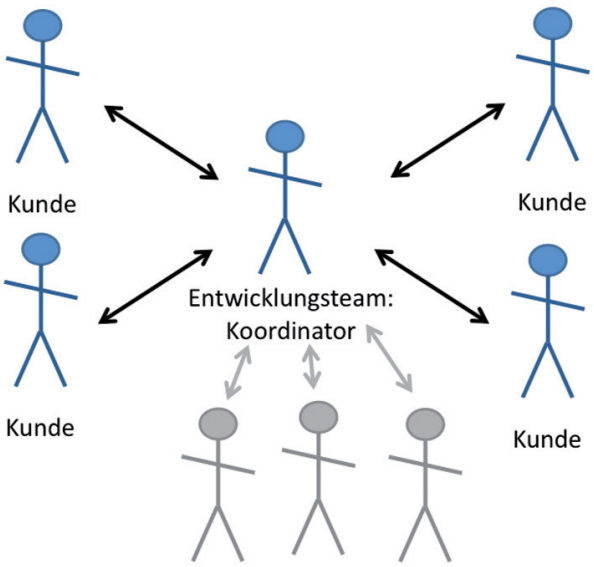

Abb. 3: Spezialisierung im Entwicklungsteam

und Qualifikationen identifizieren und kann so auf einer feineren Granularitätsebene Ressourcen aufbauen und einsetzen.

Die Arbeitsschritte benötigen klar definierten In- und Output und man kann diese als „Services", Entwicklungsdienstleistungen innerhalb der agilen Entwicklungsorganisation, ansehen. Eine Service-Nutzung erhöht typischerweise den Aufwand für den Service-Nutzer, den man durch effizientere Ausführung der spezialisierten Teilaufgaben beim Service-Erbringer auszugleichen versucht.

Die Softwarearchitektur spielt bei der Identifikation möglichst unabhängiger Teilaufgaben eine entscheidende Rolle. Viele Aktivitäten von Spezifikation bis zum Testen drehen sich um bestimmte Komponenten oder das Zusammenspiel mehrerer Komponenten. Das Wissen der einzelnen Teammitglieder beschränkt sich dann auf bestimmte Tätigkeiten und Komponenten (Persistenz und Datenbank, GUI oder Geschäftslogik), wodurch eine vereinfachte Einarbeitung und damit leichtere Skalierbarkeit erreicht wird.

Auf Systemebene müssen alle Teilaufgaben zusammenpassen. Zunächst werden Aufgaben gemäß der Architektur heruntergebrochen und abschließend auch wieder zusammengefügt. Diese Koordinationsaufgabe wird typischerweise von einem Architekten erbracht und ist der Preis für die verbesserte Skalierbarkeit durch Arbeitsteilung.

Allerdings wird diese Koordinationsaufgabe an der Kundenschnittstelle erbracht, an der ja auch nach wie vor Agilität gefordert ist und dort durch die Koordinierenden auch gelebt wird. Die interne Arbeitsaufteilung ist generell transparent für den Kunden, er interagiert nur mit den koordinierenden Rollen.

Produktorganisation: externe und interne Agilität

Im nächsten Schritt lagert man die Teile in eigene Teams aus. Diese Teams sind nun für die Art der Ausführung ihres Arbeitsschrittes voll und ganz verantwortlich (s. Abbildung 4). Im Sinne eines Lean-Ansatzes können sie lokale Optimierungen direkt und eigenmächtig umsetzen. Die Kapazitätsplanung und Mechanismen zur dynamischen Anpassungen werden ebenfalls durch die jeweiligen Teams identifiziert und realisiert.

An den Schnittstellen in der zweiten Reihe gelten nun die gleichen Agilitätsbetrachtungen wie im klassischen Fall. Die Arbeiten beziehen sich lediglich nur noch auf ein Teilsystem. Der Grad der Agilität an allen internen Schnittstellen hat dann direkte Auswirkungen auf die erreichbare Agilität an der entscheidenden Schnittstelle zum externen Kunden.

Dies ist keine neue Erkenntnis, sondern es ist seit langem bekannt, dass die größte Herausforderung von agilen Methoden die Schnittstellen zu anderen Organisationseinheiten sind. Die gewählten Taktraten müssen aufeinander abgestimmt sein, sonst kommt es schnell zu Verzögerungen und man erreicht nicht die angestrebte Agilität an den externen Schnittstellen.

Innerhalb der eigenen Organisation kann man noch Justierungen vornehmen, geht es jedoch zum Beispiel um andere Zulieferer des Kunden, mit deren System man Schnittstellen hat, muss man sich um diese asynchrone Kooperation explizit kümmern und deren Auswirkungen berücksichtigen. Eine solche Schnittstelle kann auch den Versuch eines agilen Vorgehens generell massiv stören, wenn man z. B. nur einmal eine Schnittstelle zu einem Drittsystem bestellen kann und deshalb schon vorausschauend vieles potenziell benötigte früh fordern muss.

Letztendlich trifft auch der klassische Fall in der Praxis häufig bereits auf einen solchen Kontext, ohne dass man das Entwicklungsteam in Services und kleinere

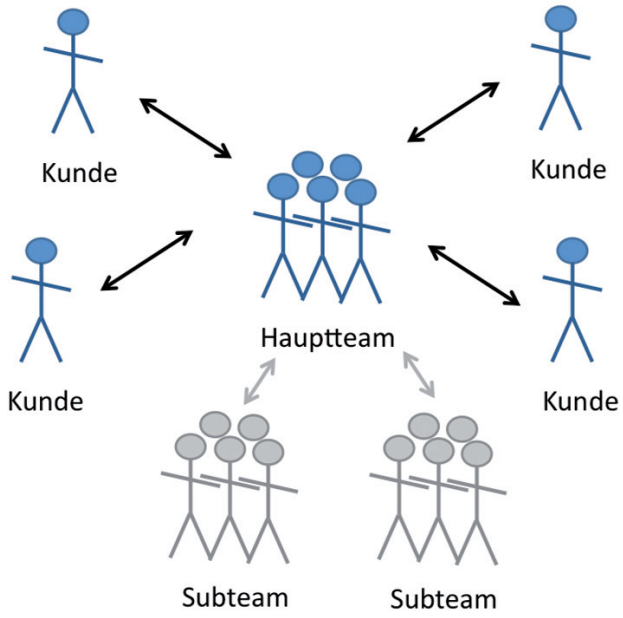

Abb. 4: Produktorganisation mit spezialisierten Subteams

Teams unterteilt. Aus diesem Grund ist stets eine organisatorische Analyse notwendig, um die erreichbare Agilität zu charakterisieren und mit dem Kunden zu besprechen. Es muss stets dafür gesorgt werden, dass man an der Kundenschnittstelle aktiv Problemfelder anspricht und nach Lösungen sucht.

Für Neukunden gilt, dass man nicht bei null beginnt, sondern initial das vorhandene Produkt und dessen Konfigurationsmöglichkeiten ausnutzt, um schnell Fortschritte zu erzielen; auch dies kann zur empfundenen Agilität positiv beitragen. Ein wichtiger Nebeneffekt ist dabei, den Kunden rasch an das Produkt heranzuführen und ihn dazu zu bringen, seine Anpassungswünsche relativ zum bestehenden Produkt zu formulieren. Dies vereinfacht, Anforderungen zu verstehen, als Teil des Produkts zu sehen und entsprechend umzusetzen.

Service-basierte Produktion: Optimierung vieler Produkte

Produziert eine Organisation mehr als ein einzelnes Produkt, stellt sich schnell die Frage, inwieweit die Arbeitsteilung innerhalb eines Produktes auch für andere Produkte passt (siehe Abbildung 5). Dies hängt direkt mit der Homogenität der eingesetzten Technologien, Tools und Prozesse zusammen. Standardisierung ist hier ein Weg, eine ausreichende Homogenität zu erzielen, die gute Optimierungsmöglichkeiten bietet.

Unterm Strich ist das zu schaffende Optimierungspotenzial über viele Produkte grundsätzlich eines der Hauptargumente überhaupt für die Standardisierung. Dienstleistungen wie Optimierungen oder Versionsupgrades von Datenbanken oder Application Servern sind Beispiele, von denen viele Produkte profitieren können. Referenzarchitekturen sorgen dafür, dass Systemteile auf die gleiche Art und Weise interagieren.

Auch dies bietet großes Potenzial für gemeinsame Basisfunktionalität und Realisierungsmuster: Experten können optimal eingesetzt und ausgelastet werden, die Bilanz für Prozessverbesserungen oder -automatisierungen sieht auch deutlich positiver aus.

Die große Herausforderung besteht darin, das Gesamtsystem adäquat zu definieren und die kontinuierliche Verbesserung und Optimierung am Leben zu erhalten. Grundsätzlich gilt aber für alle Einzeldienste die analoge Diskussion eines Einzelteams, welches Agilität mit mehreren

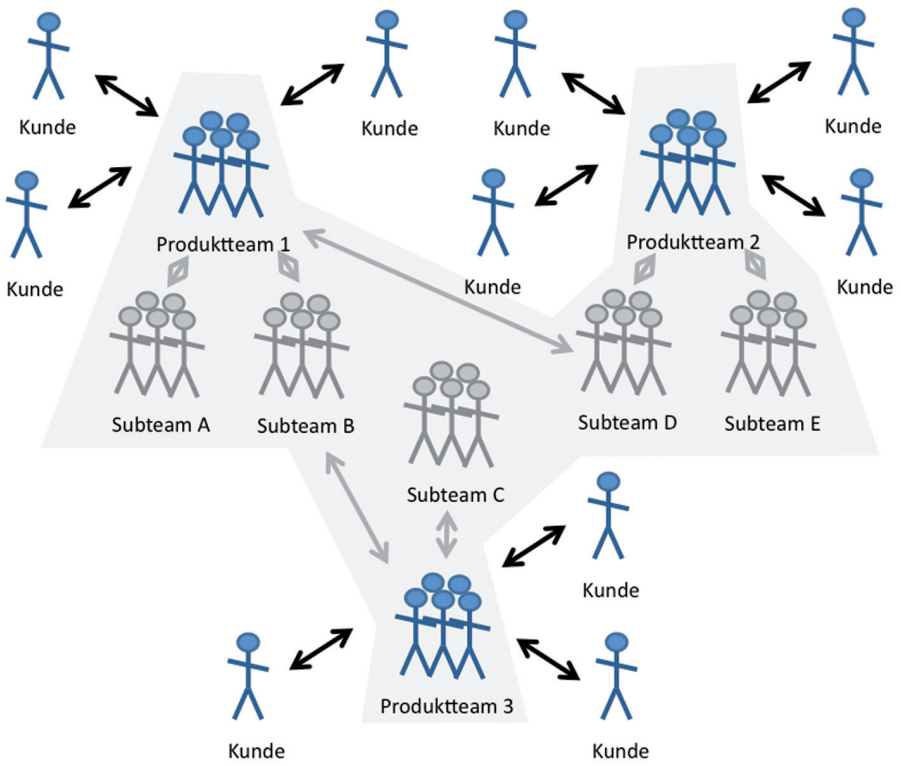

Abb. 5: Produktübergreifende Entwicklungsorganisation

Kunden erreicht. Im internen Fall sind die Kunden verschiedene Produktteams. Das Service-Paradigma erlaubt demzufolge, weitere Schichten hinzuzufügen und das Gesamtgebilde stets den Bedürfnissen anzupassen.

So kann ein Projekt, das mehrere Produkte bei einem Kunden integriert und einführt, als Kunde für die Einzelprodukte agieren. Die Kundenschnittstelle ist das integrierende Team, die einzelnen Produktteams sind im Hintergrund und für den Kunden weitestgehend transparent. Dies erlaubt ein einheitliches und konsistentes Kommunikationsverhalten mit dem Kunden über alle Aspekte des integrierten Systems. Ein weiterer Baustein, der Agilität an der Kundenschnittstelle unterstützt.

Fazit

Agile Methoden sind mittlerweile schon längst im Mainstream angekommen – zu Recht. Es gibt praktisch niemanden mehr, ob Manager oder Entwickler, der von diesen Methoden noch nichts gehört hat. Viele davon fordern auch stetig ein, dass man „agiler werden muss“ und stehen demzufolge der Einführung von agilen Methoden offen gegenüber.

Dies führt zu der eher seltenen Konstellation, dass Änderungen von Prozessen und Praktiken in der Entwicklung leichter geduldet werden und man die in der Entwicklung „einfach mal“ machen lässt ohne die sonst üblichen Bedenken und endlosen Listen von identifizierten Knackpunkten. Bevor man jedoch an einer Stelle agile Methoden einführt, ist eine gesamtorganisatorische Analyse nicht zu vermeiden, Anpassungen sind notwendig, um das gewünschte zu erreichen. Agilität sorgt nicht automatisch auch für minimale Kosten, man muss genau abwägen, welche Vorteile man dadurch bekommt und was diese einem selbst bzw. dem Kunden Wert

Referenzen

[CarO8] Ralf Carbon, Jens Knodel, Dirk Muthig: Providing feedback from application to family engineering – the product line planning game to the Testo AG. Software Product Line Conference, 2008.

[Lin04] Mikael Lindvall, Dirk Muthig et. al.: Agile software development in large organizations. IEEE Computer, No.12, 2004.

[Kle04] Marco Klemm, Dirk Muthig: Kostenoptimierung durch agile Methoden und Produktlinien, OBJEKTspektrum, Nr.5, 2004.

[And10] David Andersen: Successful Evolutionary Change for Your Technology Business, BlueHole Press, 2010.