März 2014

TIME-TO-MARKET DURCH WIEDERVERWENDUNG:

Vom Anforderungsmanagement zur fachlichen Architektur

Juri Urbainczyk
Agon Solutions

Abstract

Bei der Erschließung neuer Geschäftsmodelle und bei der Sicherung bekannter Märkte nimmt die IT eine Schlüsselrolle ein. Eine der häufigsten Anforderungen in diesem Kontext heißt Time-to-Market. Es ist die Fähigkeit, neue Angebote und bessere Produkte rechtzeitig zum Kunden zu bringen. Wer jedes Mal das Rad neu erfindet, wird nie eine gute Time-to-Market erreichen: Wiederverwendung ist also gefragt und wichtiger denn je. Woran liegt es, dass Wiederverwendung in Softwareprojekten zwar oft beschworen, aber nur selten realisiert wird? Sowohl die Organisation des Projekts und des Unternehmens als auch die Softwarearchitektur spielen hier eine wichtige Rolle. Aber selbst die beste Architektur führt nicht von selbst zu einem hohen Grad an Wiederverwendung, wenn das Anforderungsmanagement dem nicht Rechnung trägt. Dieser Artikel geht der Frage nach, was zu tun ist, um effektive Wiederverwendung zu erreichen, und was das Anforderungsmanagement dafür leisten kann.

Warum wiederverwenden?

Das Bedürfnis nach der geringsten Time-to-Market fordert Softwareentwickler immer wieder heraus: Neue Features müssen schnell umgesetzt und Änderungen rechtzeitig mit hoher Qualität durchgeführt werden. Wie ist das zu schaffen? Der gesamte Entwicklungsprozess muss auf den Prüfstand, um kurze Turnaround-Zeiten sicherzustellen. Eine hohe Automatisierung von Test und Deployment sind dazu nötig. Was aber kann in der Entwicklung selbst getan werden? Eine Möglichkeit ist Konfiguration: Die Anwender können ihre Wünsche in einer bequemen GUI mit wenigen Mausklicks umsetzen. Sind die Änderungen jedoch tiefgreifend und umfassen Daten, Algorithmen und Abläufe, so führt eine Entwicklung per Konfiguration schnell zu einer übermäßig komplexen und bald nicht mehr wartbaren Anwendung – denn jede einzelne Konfigurationsmöglichkeit muss in der Implementierung der Anwendung vorweggenommen werden.

Wesentlich effizienter lassen sich kurze Entwicklungszyklen erreichen, indem man bereits vorhandene und bewährte Bausteine erneut einsetzt. Auch Autos lassen sich nur dann schnell entwickeln und bauen, wenn man bekannte Elemente – wie z.B. die sprichwörtlichen Räder – nicht jedes Mal neu erfindet. Somit ist Wiederverwendung eine wichtige Grundlage, um in der Entwicklung schnell auf neue Anforderungen reagieren zu können, d.h. eine wesentliche Voraussetzung für Agilität. Darüber hinaus steigt die Qualität, wenn bereits getestete und produktive Komponenten erneut eingesetzt werden können. Im Umkehrschluss sinken dadurch die Kosten (unter bestimmten Voraussetzungen, später dazu mehr). Hinzu kommen die klaren Vorteile bei der Wartbarkeit, die durch den Einsatz vorhandener Komponenten erzielt werden können: Geschäftslogik ist nur noch jeweils einmal vorhanden und kann somit einfach lokalisiert und geändert werden; Änderungen an einem zentralen Baustein wirken sich konsistent auf alle Nutzer der Komponente aus.

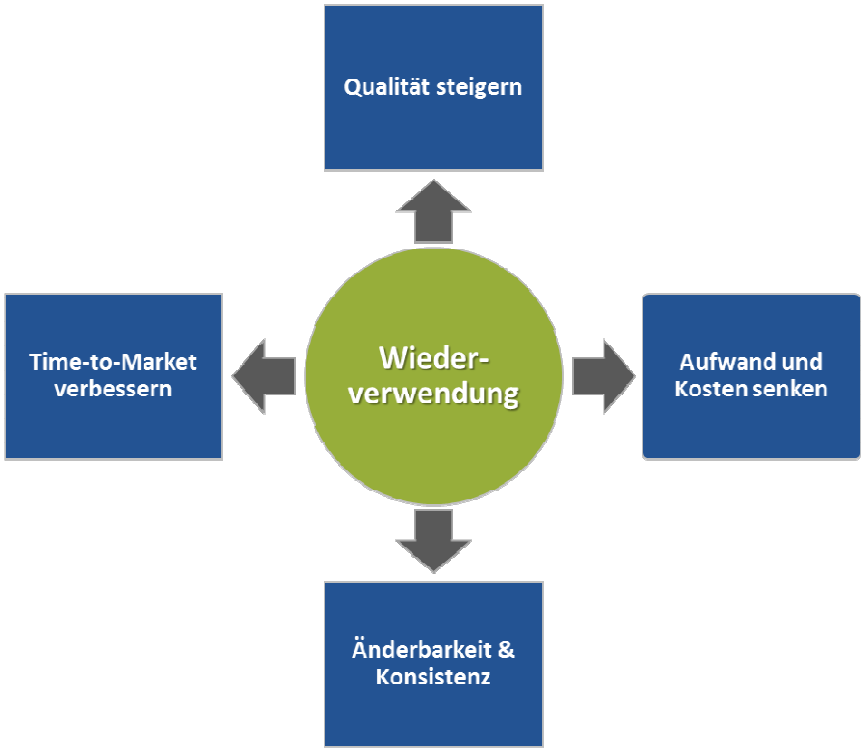

Abbildung 1: Vier gute Gründe für Wiederverwendung

Auf der Basis von wiederverwendbaren Komponenten ist es außerdem möglich, die Anwendungsentwicklung insgesamt auf eine höhere Ebene zu heben: Anwendungen lassen sich aus vordefinierten Bausteinen zusammenbauen – Kombinieren statt Programmieren.
Wiederverwendung ist also absolut wünschenswert und es wird auch viel davon gesprochen und geschrieben. Fragt sich nur, warum sie in der Realität so selten erfolgreich betrieben wird. Daher schauen wir uns die Voraussetzungen, die für eine effektive Wiederverwendung notwendig sind, jetzt genauer an.

Voraussetzungen für Wiederverwendung

Um Wiederverwendung in einem Softwareprojekt mit Erfolg einzuführen, müssen vier Faktoren zusammenspielen: Architektur, Infrastruktur, Organisation und Anforderungsmanagement.
Die Software-Architektur muss Wiederverwendung unterstützen. D.h. es muss möglich sein, wiederverwendbare Bausteine (vulgo: „Komponenten“ oder auf anderer Aggregationsebene „Services“) zu konzipieren und zu implementieren. Die Architektur muss also den Begriff einer „Komponente“ kennen und die technischen Möglichkeiten bereitstellen, eine solche umzusetzen. Oft liefern bereits bestimmte Frameworks oder Produkte wie Portalserver oder SOA-Suiten die passenden Ansätze. Viele Initiativen zur Wiederverwendung hören an dieser Stelle auf. Leider ist es nicht ausreichend, ein entsprechendes Framework lediglich zur Verfügung zu stellen, sondern auch die Verwendung (das „Nutzungskonzept“) muss so ausgelegt sein, dass sinnvolle Komponenten ermöglicht werden.

Die Infrastruktur muss ein Repository für die Komponenten und ihre Dokumentation bereitstellen. Auch die Suche nach Komponenten muss möglich sein. Eine Ablaufumgebung für automatisierte Tests muss existieren. Oft – aber nicht immer – wird auch ein Container für Services (z.B. ein Enterprise Service Bus, o. ä.) benötigt.
Für effektive Wiederverwendung sind zudem organisatorische Aspekte zu berücksichtigen. Da der Wert der Wiederverwendung sich meist später zeigt als die dadurch ohne Zweifel entstehenden Aufwände, ist Durchhaltevermögen gefordert: Immer wieder müssen die guten Gründe für dieses Vorgehen aufgezeigt werden, um die Motivation im Team und bei den Stakeholdern zu erhalten. Zudem benötigt man organisatorische Rollen wie Komponenten-Bereitsteller und Komponenten-Konsumenten, die bekannt sein und gelebt werden müssen. Insbesondere die Verantwortung für einzelne Komponenten (u.a. für Rückfragen und Weiterentwicklung) muss eindeutig geklärt sein.
Das Anforderungsmanagement muss die Wiederverwendung ebenfalls unterstützen. Dies ist ein Punkt, der häufig unterschätzt wird und nach Erfahrung des Autors einer der Hauptgründe für fehlende bzw. wenig effektive Wiederverwendung ist. Tatsächlich ist fachliches Domänenwissen notwendig, um sinnvoll wiederverwendbare fachliche Komponenten zu erstellen.
Im Folgenden gehen wir optimistisch davon aus, dass die ersten drei Voraussetzungen (in Architektur, Infrastruktur und Organisation) erfüllt sind, und konzentrieren uns darauf, warum vor allem Anforderungsmanagement wichtig für die Wiederverwendung ist. Das wird deutlicher, wenn man sich die Eigenschaften der Komponenten anschaut.

Eigenschaften von Komponenten

Über die Frage, was genau Komponenten sind, ist schon reichlich gestritten worden. Daher beschränken wir uns hier auf die Eigenschaften einer Komponente, die wesentlich für ihre Wiederverwendbarkeit sind:

1. ***Definition:*** *Eine klar definierte und dokumentierte Verantwortlichkeit*
2. ***Isolation:*** *Unabhängigkeit von anderen Artefakten*
3. ***Kombination:*** *Gute Einsetzbarkeit in einem anderen Kontext als dem ursprünglichen*

Es ist für diese Betrachtung nicht wesentlich, von welcher *technischen* Ausprägung einer Komponente gesprochen wird. Es kann sich um Java-Packages, Portlets, SOA Services, EJB oder OSGI-Module handeln – die oben genannten Eigenschaften gelten in jedem Fall. Schauen wir uns die einzelnen Eigenschaften nun der Reihe nach an. Warum sind sie wirklich notwendig und warum ist Anforderungsmanagement dafür wichtig?

Definition

Wenn ein Entwickler eine bereits existierende Komponente einsetzen will, so muss er genau und zuverlässig wissen, ob und wobei sie ihm helfen kann. Denn nur so kann er sicher sein, dass er mit dieser – ihm ja zunächst fremden – Komponente tatsächlich sein vorgegebenes Ziel erreicht. Der Faktor Motivation ist nicht zu unterschätzen: Entwickler werden nur dann den Aufwand treiben, sich die Komponente genauer anzusehen, wenn sie von den Vorteilen überzeugt sind. Auch müssen Aufwand und erwarteter Mehrwert in einem sinnvollen Verhältnis stehen. Der Aufwand für die Suche und für das Verstehen der Komponente muss sich in Grenzen halten. Daher sollte ein Katalog von Komponenten vorliegen, aus dem man sich bedienen kann, wie aus einem Werkzeugkasten.

Erst wenn bekannt ist, was bestimmte Teile eines Systems tun, ist es möglich zu erkennen, dass genau dasselbe – oder zumindest etwas sehr ähnliches – noch ein weiteres Mal gebaut werden soll. Dazu braucht der Nutzer nicht zu wissen, „wie“ die Komponente ihre Dienste erbringt; es reicht völlig aus, zu wissen „was“ sie tut und über welche Schnittstellen sie zu bedienen ist. Die Verantwortung, welche der Komponente aus fachlicher Sicht zugewiesen wird, muss eindeutig benannt sein. Unterschiedliche, noch dazu fachlich zusammenhanglose Aufgaben sollte man nicht mischen. Es ist wichtig eine vermittelbare Idee zu entwickeln, für welche die Komponente steht. Dafür ist es nützlich, saubere Begrifflichkeiten zu definieren und ein klares Verständnis zu erarbeiten. Dies alles gehört zur fachlichen bzw. funktionalen Beschreibung und ist eine typische Aufgabe des Anforderungsmanagements.

Isolation

Wenn eine Komponente wiederverwendet werden soll, geschieht dies meist in einem neuen Kontext, der von dem Kontext ihrer Erstellung abweicht. Daher muss die Komponente möglichst isoliert arbeiten, ohne bestimmte Annahmen über andere Teile des Systems zu treffen. Problematisch sind vor allem die impliziten Kontexteigenschaften, deren sich die Fachseite oft nicht bewusst ist. Abhängigkeiten zwischen verschiedenen „Parametern“ werden gern übersehen. Falls dennoch Abhängigkeiten existieren (was sich nicht immer vermeiden lässt), müssen diese dokumentiert und bekannt sein. Diese Aufgabe muss primär das Anforderungsmanagement leisten, denn es handelt sich meist um fachliche Abhängigkeiten. Eine im Sinne der Wiederverwendbarkeit „gute“ Komponente hat also möglichst wenige Abhängigkeiten, die vollständig dokumentiert sind.

Kombination

Wiederverwendung geschieht nie im exakt selben Kontext wie der Ersteinsatz. Z.B. können Daten zum Füllen eines Parameters fehlen. Ist eine Komponente unter geänderten Bedingungen nicht einsetzbar, ist die Kombinierbarkeit mangelhaft und eine Wiederverwendung nicht möglich. Dazu folgt ein (vereinfachtes) Beispiel:

Nehmen wir an, unser Entwickler hat einen Service implementiert, der Flugtarife aus einer Backend-Datenbank sucht. Der Service erwartet u.a. die zwei Parameter Datum des Abflugs und User-ID. Die User-ID wird benötigt, da an der Stelle des ersten Einsatzes immer ein angemeldeter Benutzer bekannt ist, für den die Suche nach den Tarifen personalisiert werden soll. Im Fall der Wiederverwendung liegt kein Benutzer vor, da der Service an dieser

Stelle in einem anderen Kontext aufgerufen wird. Wenn die User-ID ein Muss-Parameter ist, scheitert die Wiederverwendung schon in diesem frühen Stadium. Um dieses Problem zu umgehen, könnte der Entwickler z.B. in der Implementierung prüfen, ob die User-ID null oder leer ist und dann die Personalisierung nicht aufrufen. Der Parameter könnte als „optional“ dokumentiert werden. Diese geringfügige Änderung erhöht das Potential für die Wiederverwendung enorm. Der Entwickler kann das noch verbessern und weitere Parameter optional machen, indem er sinnvolle Defaultwerte vorsieht; z.B. könnte er für das Datum des Abflugs den jeweils nächsten Tag einsetzen.
Dieses Beispiel zeigt, dass schon einfache Verfahren wie z.B. die Generalisierung der Schnittstelle viel zur Wiederverwendbarkeit beitragen. Der Entwickler und der Architekt müssen jedoch Mehrarbeit leisten. Und nicht nur das: sie müssen zuallererst davon ausgehen, dass Wiederverwendung denkbar und sinnvoll ist. Das ist nicht immer der Fall und auch nicht immer wünschenswert: Will der Auftraggeber tatsächlich, dass in jedem Fall dieser Mehraufwand geleistet wird? Es wird also deutlich, dass ein gesteuerter Prozess benötigt wird, um zu entscheiden, ob eine Komponente wiederverwendbar entworfen und gebaut wird. Im Falle fachlicher Funktionalität kann diese Entscheidung nur im Anforderungsmanagement sinnvoll getroffen werden.

Wie man zu Komponenten kommt

Alle wesentlichen Eigenschaften von Komponenten hängen also vom Anforderungsmanagement ab. Das gilt umso mehr für den Prozess, der zu wiederverwendbaren Komponenten führt.
Es reicht nicht, bestimmte mehr oder weniger zusammenhängende Artefakte aus dem Softwaresystem zu isolieren und zu Komponenten zu erklären. Das klingt lustig, kommt in realen Projekten jedoch nicht selten vor. Solche Komponenten „qua Benennung“ machen sich zwar optisch gut in einem PowerPoint-Vortrag, helfen bei der Wiederverwendung aber meist nicht weiter. Im Gegenteil, denn sie erfüllen die o.g. Bedingungen nicht: Sie sind nicht abgegrenzt, denn fast immer besitzen sie bekannte (und auch viele unbekannte) Abhängigkeiten zu anderen Systemteilen. Schlimmer noch, meist sind eng zusammenhängende Funktionen über mehrere dieser Pseudo-Komponenten verteilt, so dass sie nicht sinnvoll einzeln einsetzbar sind. Darüber hinaus sind sie auch nicht gut kombinierbar, da ihre Schnittstellen nicht dokumentiert und nur auf einen speziellen Zweck abgestimmt sind. Will man sie mit anderen als den bisherigen „Partnern“ kombinieren, müssen beide Systemteile umfangreich modifiziert werden. Last but not least: Ihre Aufgaben sind nicht klar umrissen und schon gar nicht beschrieben, sie haben also keinen vermittelbaren Sinn. Meist hat ein Entwickler ad-hoc entschieden, was die jeweiligen Zeilen Quelltext bewirken sollen. Mit etwas Glück arbeitet dieser Entwickler noch für das Unternehmen – wenn nicht: Game Over. Da man etwas, das nicht beschrieben ist, nicht wiederverwenden kann, baut man häufig dasselbe (oder eine geringfügig modifizierte Variante dessen) noch einmal.

Die Ausrufung von „Komponenten“ im Nachhinein ist mindestens schwierig, wenn nicht unmöglich. Wiederverwendbarer Bausteine entstehen nicht automatisch (sozusagen „spontan“), sondern nur durch einen definierten und sinnvollen *Prozess*. Der erste Schritt ist es, zunächst zu erkennen, dass sich eine bestimmte Funktion zur Wiederverwendung eignet oder dass sich für ihre Umsetzungen bereits vorhandene Bausteine verwenden lassen.
Was ist eigentlich sinnvoll wiederverwendbar? Das können nur solche Funktionen oder Services sein, die mehrfach an verschiedenen Stellen im System verwendet werden (sollen). Wie identifizieren wir diese Funktionen?
Einem Entwickler fallen zunächst gerne technische Querschnittsfunktionen (z.B. Logging oder Tracing) ein. Und tatsächlich sollten genau solche technischen Komponenten existieren und durchgängig und einheitlich genutzt werden. Technische Funktionen sind jedoch nicht entscheidend, um schnell neue fachliche Funktionen umzusetzen. Denn meist liegt der Löwenanteil der Implementierungsaufgaben bei der Geschäftslogik. Und genau diese gilt es so zu implementieren, dass sie wiederverwendet werden kann.
Welche Teile der Geschäftslogik am besten wiederzuverwenden sind, hängt von der jeweiligen fachlichen Domäne ab. Hier ein kleines Beispiel: Entwickelt das Projekt gerade ein Portal für Automotive-Zulieferer, werden Funktionen wie „Zollerklärung abgeben“ oder „Zertifizierung bestätigen“ für nahezu jeden Zulieferer relevant sein. Daher ist es sinnvoll, genau diese Funktionen wiederverwendbar zu gestalten.
Nach der Erfahrung des Autors geschieht genau das leider nicht. Typischerweise wird zunächst eine Anwendung für Zulieferer von Sitzbezügen implementiert, welche bereits „Zollerklärung abgeben“ und „Zertifizierung bestätigen“ enthält. Einige Zeit später wird dann eine analoge Anwendung für die Elektronik-Zulieferer gebaut, die ebenfalls erneut beide Funktionen abbildet. Wieder etwas später kommen dann die Zulieferer für IT an die Reihe und erneut werden die genannten Funktionen separat implementiert; jedes Mal ein wenig anders und spezifisch auf die jeweiligen Datenobjekte ausgerichtet. Schließlich werden alle drei Anwendungen in das Portal integriert und die Fachabteilung wundert sich, warum bei einer Änderung an der Geschäftslogik zur Zollerklärung alle Anwendungen geändert werden müssen – was natürlich sowohl Aufwand als auch Kosten in die Höhe schraubt (während die Qualität sinkt). Eine gute Time-to-Market ist so nicht zu erreichen.
In diesem Beispiel wurde offensichtlich nicht wiederverwendet. Und das obwohl Softwarearchitektur, Infrastruktur und Organisation dazu möglicherweise in der Lage gewesen wären. Denn diese Faktoren sind zwar notwendig, aber nicht hinreichend. Zusätzlich ist auch ein *durchgängiges Anforderungsmanagement* nötig. Der Anforderungsmanager, der die Fachlichkeit kennt und einen Überblick über vorhandene und geforderte Funktionen hat, kann erkennen, dass eine adäquate Komponente bereits vorhanden ist und dass es sinnvoll ist, sie wiederverwendbar zu gestalten.

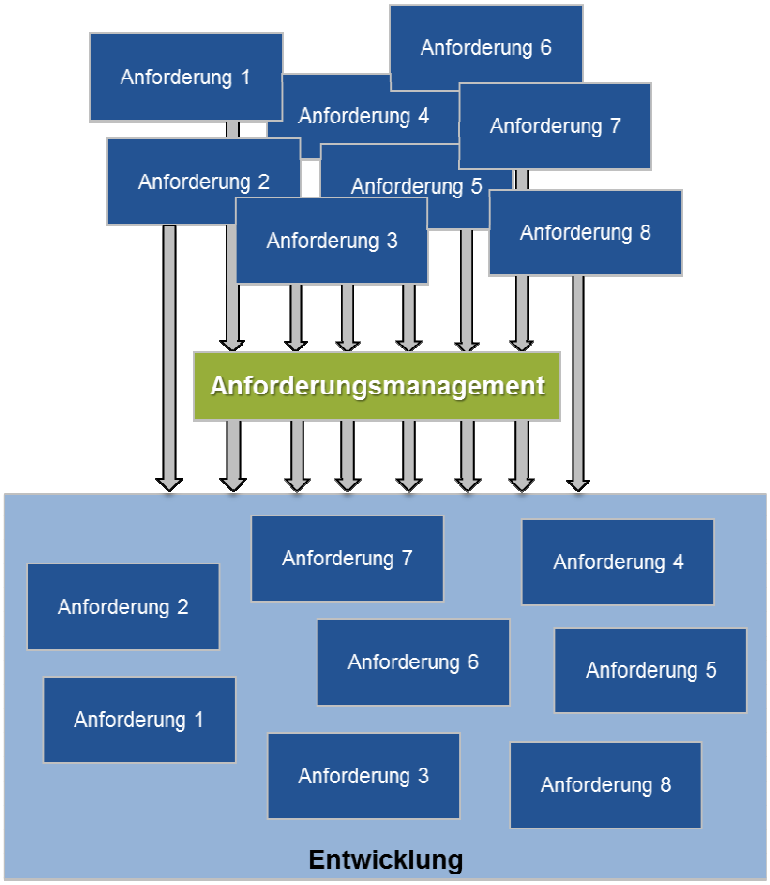

Abbildung 2: Ungeordnete Anforderungen

Komponenten schneiden

Anforderungsmanagement ist also notwendig, um wiederverwendbare Funktionen zu erkennen und die notwendigen Eigenschaften der Komponenten herbeizuführen. Dazu muss im Anforderungsmanagement ein geordnetes Mapping von Anforderungen auf Komponenten durchgeführt werden. Geschieht das nicht, erreichen die einzelnen Anforderungen ungeordnet die Entwicklung und können dort nicht mehr zu sinnvollen Komponenten zusammengefügt werden (s. Abbildung 2).

Dazu ein Bespiel: Nehmen wir an, es gibt die drei Anforderungen, „(a) Speichere die Adresse“, „(b) Erfasse die Versandanschrift“ und „(c) Erfasse die Artikelnummer“. Erreichen diese Anforderungen die Entwicklung in ungeordneter Reihenfolge (eventuell sogar mit großem zeitlichem Abstand), ist es nicht möglich, die Anforderungen (a) und (b) in einer Komponenten „Adressmanagement“ zu kombinieren. Vielleicht werden (b) und (c) in einer Komponente „Neue UI Elemente“ zusammengeführt – aber das wäre sicher keine sinnvolle Lösung.

Das Anforderungsmanagement muss also die Anforderungen bündeln, d.h. in solche Cluster zusammenfassen, die sich für Komponenten eignen. Dazu gehört auch, zu erkennen, dass bereits eine Komponente existiert, der man eine bestimmte Anforderungen als weitere Eigenschaft zuordnen kann. Je nachdem, wie dieses Mapping ausfällt, können die

Abhängigkeiten der Komponenten ganz anders ausfallen; hier entscheidet sich ihre „Isoliertheit“. Es geht darum, fachliche Ähnlichkeiten und Zusammenhänge zu identifizieren, auch wenn sich die Anforderungen auf den ersten Blick unterscheiden. Dazu ist Abstraktionsfähigkeit und Domänenwissen notwendig. Der Komponentenschnitt ist somit eine wesentliche Aufgabe des Anforderungsmanagements und nur dort erfüllbar.
Im Idealfall werden die Anforderungen also vom Anforderungsmanagement so gebündelt, dass sie geordnet und in sinnvollen Paketen die Entwicklung erreichen (s. Abbildung 3). Dabei sollte auch bedacht werden, ob die Anforderung bzw. die damit verbundene Eigenschaft oder Funktion sich eher auf ein Datenobjekt oder auf einen Prozess bezieht. Wenn man in der Lage ist Prozesslogik von datenbezogener Logik zu trennen, wird Wiederverwendung noch einmal wirkungsvoller, da sich oft nur eines von beiden ändert und bei einer neuen Variante eines Prozesses die „alten“ Datenobjekte wiederverwendet werden können. Im Beispiel der Zollanwendung, hieße es, das Datenobjekt „Zollerklärung“ mit seiner Geschäftslogik vom Prozess „Zollerklärung abgeben“ zu separieren. Das ergäbe dann zwei getrennte Komponenten.

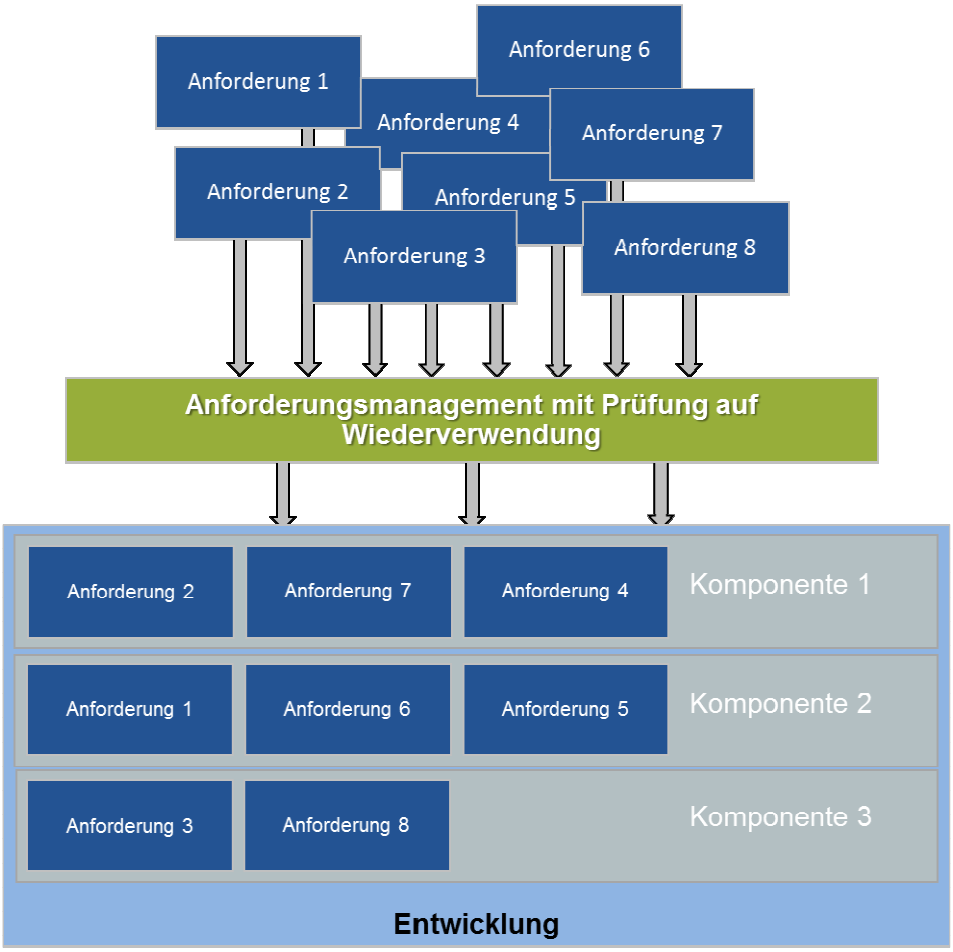

Abbildung 3: Anforderungen mit Wiederverwendbarkeit

Fachliche Architektur für Wiederverwendung

Es ist klar geworden, dass ein gesteuerter Prozess für Wiederverwendung sorgen muss, in dem Anforderungsmanagement eine wesentliche Rolle spielt. Dabei hat der Anforderungsmanager die folgenden Aufgaben:

a) Die Funktionen und fachlichen Inhalte der Komponenten müssen definiert werden. Eine wesentliche Herausforderung ist es, die meist ungeordnet eintreffenden Anforderungen so zu bündeln und zu strukturieren, dass sie in verständliche und sinnvolle Komponenten überführbar sind. Darüber hinaus muss der Anforderungsmanager die Anforderungen benennen, welche in erster Linie von Wiederverwendung profitieren und welche sich zur Erstellung wiederverwendbarer Komponenten eignen.

b) Ein Katalog an fachlichen Funktionen und Komponenten ist zu erstellen. Der Katalog hilft, Komponenten schnell zu finden (und so zu verwenden) und unterstützt bei der Konzeption neuer Komponenten. Ein Glossar zur begrifflichen Abgrenzung ist eine sinnvolle Ergänzung.

c) Die fachlichen Abhängigkeiten zwischen Komponenten sind zu erkennen und zu minimieren. Je nachdem, wie Komponenten geschnitten werden, können Abhängigkeitsgrafen unterschiedlicher Komplexität entstehen (mit entsprechenden Auswirkungen für die Wartbarkeit).

d) Die fachlichen Schnittstellen einer Komponente müssen entworfen werden. Dabei muss (wie im obigen Beispiel) die Wiederverwendung berücksichtigt werden, d.h. die Schnittstellen müssen so spezifiziert werden, dass sie unter beliebigen Bedingungen einsetzbar sind.

Alle diese Aufgaben sind unabhängig von Technik und von spezifischen Frameworks, und daher können und müssen sie bereits im Anforderungsmanagement effektiv gelöst werden. Der Anforderungsmanager muss dazu mehr tun als rein verbal fachliche Anforderungen zu paraphrasieren. Es sind Komponenten zu beschreiben und Schnittstellen zu designen. Der Begriff „Fachlicher Architekt“ ist für solche Aufgaben sicherlich gerechtfertigt. *Um effektive Wiederverwendung zu erreichen, muss das Anforderungsmanagement als eine Architekturaufgabe betrachtet werden.* Der Ansatz „Ich schreibe mal auf, was der Auftraggeber mir gesagt hat“ ist nicht zielführend. Echte Konzeptionsarbeit an der fachlichen Architektur ist nötig. Wer diese Herausforderung scheut, hat bei der Time-to-Market keine Chance.

Zusätzlich müssen die Rahmenbedingungen stimmen, um eine fachliche Architektur zu ermöglichen. Die Anforderungsmanager müssen langfristig an einem Projekt arbeiten können. Sie müssen die fachliche Domäne beherrschen und mit der notwendigen Entscheidungsbefugnis ausgestattet sein. Oft genug entscheidet der Auftraggeber spontan, in welchen Sprint eine Anforderung übernommen wird. Dadurch wird eine effektive Komponentenbildung erschwert, denn gerade die Bündelung von Anforderungen ist (wie oben erläutert) eine wichtige Aufgabe bei der fachlichen Architektur. Wiederverwendung erfordert also auch manchmal, einen Konflikt mit dem Auftraggeber auszuhalten und auf die mittel- bis langfristigen Mehrwerte hinzuweisen.

Fazit

Wer Wiederverwendung wünscht, muss das Anforderungsmanagement stärken. Insbesondere bei verteilten Projekten oder auch bei Portalsystemen wird eine zentrale Stelle benötigt, welche die Anforderungen auf ihr Potential prüft, wiederverwendet zu werden oder andere, bereits vorhandene Komponenten zu nutzen. Je größer ein Projekt, umso mehr Potential für Wiederverwendung liegt bei den rein fachlichen Funktionen.

Notwendige Voraussetzungen für das Gelingen von Wiederverwendung sind eine entsprechende Softwarearchitektur, Organisation und Infrastruktur. Das allein aber reicht nicht. Um Anforderungen zu sinnvollen Komponenten zu bündeln, ist fachliche Architektur gefragt. Hierzu wird ein passender Prozess mit geeigneten Verantwortlichkeiten benötigt. Wiederverwendung gibt es also nicht zum Nulltarif. Mit den richtigen Voraussetzungen ist der mögliche Gewinn für die Time-to-Market und die Qualität immens und daher jede Anstrengung wert.

Der Autor

Juri Urbainczyk ist Bereichsleiter und IT-Berater bei der Agon Solutions in Eschborn. Seit mehr als 18 Jahren befasst er sich mit Webtechnologien und Softwarearchitekturen. Herr Urbainczyk konzipiert und entwickelt Portalsysteme und mobile Anwendungen für Kunden im Bereich Aviation, Versicherungen, Banken und Logistik. Weitere Schwerpunkte sind IT-Audits und Enterprise Architektur. Herr Urbainczyk lebt mit seiner Familie in der Wetterau.

Agon Solutions

Die Agon Solutions, 2004 gegründet, ist ein unabhängiges IT-Dienstleistungsunternehmen mit Firmensitz in Eschborn bei Frankfurt und weiteren Standorten in Hamburg und Berlin. Das branchenübergreifende Dienstleistungsportfolio von Agon umfasst das „Agon-proven IT-Consulting“, eine bewährte, herstellerneutrale IT-Beratung; sowie die „Agon-tailored IT-Solutions“, zu denen maßgeschneiderte, individuelle Softwareentwicklung und passgenaue Softwareintegration gehören. Agon Solutions stellt sich individuell auf seine Kunden ein und setzt auf professionelles Projektmanagement, proaktives Anforderungsmanagement sowie änderbare Softwarearchitekturen („Design for Change“) – für Ergebnisse von besonders hoher Qualität. Die Stärken liegen in der Integration und Migration von Systemen, der Vernetzung von Mainframe- und Web-Anwendungen und der Architekturberatung von Software-Landschaften. In ausgewählten Branchen wie Banken, Versicherungen, Touristik/Aviation und Health Care bietet Agon Solutions gemeinsam mit seinen Partnern Lösungen, die auf fundiertem Geschäftsprozess-Know-how beruhen und speziell auf die jeweilige Branche zugeschnitten sind: die „Agon-tailored Business Solutions“. Die plattformübergreifende technologische Kompetenz bei Agon Solutions reicht von klassischen Mainframe-Architekturen bis hin zu modernen Java/JEE Web- und Portal-Architekturen. Zu den Referenzkunden von Agon gehören unter anderen die AOK Berlin-Brandenburg, die Commerzbank, die Deutsche Bank, die Deutsche Bank Bauspar, die Deutsche Börse, die Finanz Informatik, die Provinzial Nordwest, die Deutsche Lufthansa und die Thomas Cook.

Copyright:

A:gon Solutions GmbH

Frankfurter Strasse 71-75
D-65760 Eschborn
Telefon : +49 6196 80269 0
Telefax : +49 6196 80269 11
http://www.agon-solutions.de

Handelsregister Frankfurt HRB 58185
St.-Nr. 4022826171
Geschäftsführer: Udo Peters