Zusammenwirken von Anwendern und Softwareentwicklern bei der Abwicklung komplexer Softwareprojekte

Darlegung der Aufgaben und Pflichten, die Anwender und Entwickler bei der Zusammenarbeit im Rahmen von komplexen Softwareentwicklungsprojekten zu übernehmen haben.

Vorbemerkung:
Unter dem Begriff Anwender werden nachfolgend pauschalierend unterschiedliche Berufe/Rollen der Fachseite zusammengefasst (z.B. Manager, Auftraggeber, Organisatoren, die den Fachabteilungen zugeordnet sind, Benutzer, die mit dem neuen System arbeiten sollen, etc.). Ebenso ist der Begriff (Software-) Entwickler sehr pauschalierend. Auch hier werden sicherlich bei der Abwicklung eines Softwareentwicklungsprojektes auf der IT - Seite verschiedene Berufe/Rollen zum Einsatz kommen (z.B. Programmierer, Analytiker, Testpersonal, Projektleitungen, Netzwerkspezialisten, Datenbankspezialisten, etc.). Da hier aber speziell das (notwendige) Zusammenwirken von Mitarbeitern der Fachseite mit Mitarbeitern der IT – Seite im Mittelpunkt der Betrachtung steht, wird nachfolgend lediglich vereinfachend von Anwendern und Entwicklern gesprochen.

Bei der Abwicklung von komplexen Softwareentwicklungsprojekten ist ein Zusammenwirken von Anwendern und Entwicklern unerlässlich. Es ist nicht sehr ratsam, ein Softwarehaus mit einem komplexen IT - Projekt zu beauftragen, das Projekt mit hinreichendem Budget und mit hinreichender Zeit auszustatten und dann zu erwarten, dass nach Zeitablauf eine optimale IT – Lösung bereitgestellt wird. Eine fehlende Zusammenarbeit der Beteiligten wäh-

rend des Projektablaufs stellt ein nicht unerhebliches Projektrisiko dar.

Das Zusammenwirken von Anwendern und Entwicklern ist in der Vergangenheit nicht immer unproblematisch verlaufen. Einerseits wird von Anwenderseite beklagt, dass die Entwickler nicht mit den fachlichen Besonderheiten vertraut sind und darüber hinaus bei Argumentationen ein – für Laien – unverständliches „Fachchinesisch“ benutzen. Andererseits werden insbesondere Softwareentwickler oftmals mit der Situation konfrontiert, dass Anwender nicht exakt ihre konkreten Anforderungen formulieren können und darüber hinaus auch oft sehr abenteuerliche Vorstellungen über den mit einer Softwareentwicklung verbundenen Aufwand haben. Eine Unterschätzung des Aufwandes kommt hierbei ebenso vor, wie eine Überschätzung desselben. Weiterhin existiert bei einigen Entwicklern eine gewisse Überheblichkeit, wenn Anwender noch nicht einmal ein aus ihrer Sicht grundlegende IT – Gegebenheiten verstehen. Bei Entwicklern kommt auch noch in der Regel eine Unterschätzung der fachlichen Komplexität hinzu (Motto: Was kann denn an einer Antragsverwaltung so schwierig sein?). Diese Unterschätzung resultiert naturgemäß aus der mangelnden Kenntnis der fachlichen Gegebenheiten, die wiederum ihrerseits durch mangelnde Kommunikation der Beteiligten gefördert wird. Die Konsequenz dieser Unterschätzung der fachlichen Komplexität sind oft genug fachlich inakzeptable IT – Lösungen.

Auch bei Anwendern gab und gibt es Unterschätzungen der IT - Komplexität. Da oftmals im Privatbereich interessierte Laien kleine Applikationen eigenständig realisieren (z.B. eine Adressverwaltung mit Access) werden die Anforderungen an kommerzielle IT – Lösungen sehr leicht unterschätzt. Da man dann der Meinung ist, auch im kommerziellen Bereich schnell eine kleine Datenbanklösung ohne Einbeziehung von Entwicklern zu realisieren, ist das Resultat oftmals eine Vielzahl sog. kleiner „Insellösungen“ in Fachabteilungen, die professionellen IT - Ansprüchen in keiner Weise genügen.

Effiziente IT - Lösungen können somit nur durch sinnvolles Zusammenwirken und eine exzellente Kommunikation zwischen Anwendern (IT – Laien) und

Entwicklern (IT – Experten) geschaffen werden. Hierbei haben Anwender insbesondere bei
folgenden Aufgaben mitzuwirken und ihre Vorstellungen einzubringen:

a) Formulierung und Konkretisierung der Anforderungen
b) Lieferung des fachlichen Inputs für ein Fachkonzept und Verständnis des Fachkonzeptes
c) Mitwirkung bei der Gestaltung der grafischen Benutzeroberfläche (GUI) und von Benutzerhandbüchern
d) Lieferung von fachlichen Testfällen, an Hand derer die fachliche Korrektheit eines Systems festgestellt werden kann und Abnahme des Softwaresystems.

zu a) Formulierung und Konkretisierung der Anforderungen

Die Anforderungen an ein Softwaresystem bilden die Grundlage für alle weiteren Arbeiten. Sind die Anforderungen unvollständig, nicht konsistent oder aber unpräzise, so kommt es im weiteren Projektablauf zu häufigen Änderungen (Change Requests) oder aber nach Fertigstellung der Software zu Akzeptanzproblemen. Von Frederick Brooks stammt der 1987 geäußerte Satz „The hardest part of building a software system is deciding precisely what to build!". Dieser Satz hat auch heutzutage noch seine Berechtigung. Von Anwendern müssen somit Anforderungen formuliert werden, die das System erfüllen soll, das für sie entwickelt werden soll. Das Problem ist hierbei, dass viele Anwender nicht exakt ihre Anforderungen formulieren können. Hier müssten Entwickler unterstützend mitwirken, z.B. durch Bereitstellung von Prototypen der Benutzeroberfläche, die es dem Anwender ermöglichen, sich das neue System präziser vorzustellen. Ebenso müssten Entwickler Fragen nach den unterschiedlichen Anforderungsbereichen stellen, um Vorgaben für die Softwareentwicklung zu erhalten. Üblicherweise werden in der Praxis zunächst von Anwendern sog. funktionale Anforderungen formuliert. Dies geschieht durch Auflistung gewünschter Funktionen bzw. von Anwendungsfällen (Use

Cases). Man sollte es aber nicht bei funktionalen Anforderungen belassen. Von gleicher Wichtigkeit sind auch die nichtfunktionalen Eigenschaften einer Software. Hierzu gehören Eigenschaften wie z. B. Fehlertoleranz, Anpassbarkeit, Portabilität, Benutzbarkeit, Effizienz, Zuverlässigkeit. Diese Eigenschaften von Softwareprodukten sind in der ISO 9126 näher beschrieben. Neben funktionalen und nichtfunktionalen Eigenschaften sind aber vor Projektstart noch weitere Anforderungsbereiche zu klären. Hierzu zählt z. B. die Qualität und der Umfang der Dokumentation (Projektdokumentation. Entwicklerdokumentation, Benutzerdokumentation, Betriebsdokumentation), die Rechte an dem entwickelten Softwareprodukt, die Zahlungsmodalitäten, die (Liefer-) Termine, ggf. Konventionalstrafen, etc. Meist werden die zuletzt aufgeführten Anforderungsbereiche vor der Beauftragung eines Softwareentwicklungsprojektes nicht angesprochen und man belässt es bei den gewünschten Funktionen, den Terminen und den Budgets, eventuell wird zuvor auch noch die zukünftige Oberfläche geklärt.

Dies entspricht im weitesten Sinne den üblichen Interessen und der Vorgehensweise eines Anwenders beim Kauf eines konventionellen Produktes. Wenn man sich z. B. einen Geländewagen kauft, interessiert man sich dafür, wie er aussieht, was er leistet (Wie viel KW leistet der Motor? Kann man mit dem Wagen durch einen kleineren Fluss fahren?) und insbesondere dafür, was er kostet. Man fordert keine Bedienungsanleitung, weil man davon ausgehen kann, dass eine Anleitung im Handschuhfach liegt; man schaut sich nicht den Motor oder die Radaufhängung genauer an, weil man darauf vertraut, dass diese Teile dem neuesten technischen Stand entsprechen. Man geht auch davon aus, dass der Geländewagen bei Problemen gewartet/repariert werden kann, weil es eine technische Dokumentation gibt, in die sich auch andere Mitarbeiter gegebenenfalls einarbeiten können.

All diese Selbstverständlichkeiten sind (leider) in der Softwarebranche nicht gegeben. Deshalb ist es die Aufgabe der Entwickler, Anwender auch auf zusätzliche Anforderungsbereiche hinzuweisen, die über die Funktionalität, die Kosten und die Termine hinausgehen.

Entwickler müssten hieran insbesondere ein Interesse besitzen, wenn sie spätere Probleme verhindern wollen und an längerfristigen Kundenbeziehungen interessiert sind. Von einem Anwender/Kunden kann man die Kenntnis über die Struktur und den Umfang der Anforderungsbereiche nicht erwarten.
Anwenderseitig sind die Anforderungen allerdings nicht nur grob zu formulieren (Motto: Ich benötige ein gut benutzbares, gut dokumentiertes und funktional umfassendes und auch schnelles Softwaresystem für meine Mitarbeiter). Die formulierten Anforderungen müssen nach Möglichkeit auch so quantifiziert formuliert werden, dass eine spätere Validierung der Anforderungen möglich ist. Anforderungen sind somit als Soll - Vorgaben nicht nur für die Entwicklung, sondern auch für den Test anzusehen. Wenn man lediglich eine schnelle Software fordert, kann man im nach hinein trefflich darüber streiten, was man unter ‚schnell' versteht. Gleiches gilt auch zum Beispiel für Formulierungen, die die Dokumentation oder aber die geforderte Zuverlässigkeit angehen. Weiter ist anzumerken, dass die Anforderungen schriftlich – z.B. in einem Pflichtenheft – fest zu halten sind.
Ergänzend soll noch vermerkt werden, dass es sehr nützlich ist, zu Beginn eines Entwicklungsprojektes auch explizit gemeinsam festzulegen, welche Aufgaben/Anforderungen nicht realisiert werden sollen. Dies verhindert spätere Auseinandersetzungen um sog. graue Funktionalitäten, d.h. um Dinge, die eine Seite für so selbstverständlich hält, dass sie sie nicht erwähnt, die Gegenseite ihrerseits aber davon ausgeht, dass nur das geleistet wird, was auch explizit vereinbart worden ist. Weiterhin ist es sinnvoll, auf zu beachtende Restriktionen hinzuweisen. Hierzu können Verpflichtungen gehören, ein bestimmtes Betriebs- oder Datenbanksystem zu unterstützen, in einer festgelegten Programmiersprache zu entwickeln oder sich an Standards (z.B. Oberflächenstandards, Sicherheitsstandards, Programmier-Styleguides, etc.) zu halten.

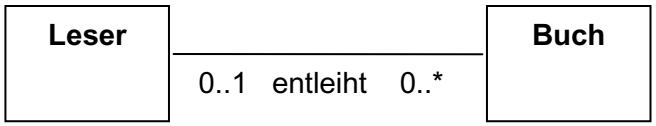
zu b) Mitwirkung, Lieferung des fachlichen Inputs und Verständnis eines sog. Fachkonzeptes

Auf der Grundlage eines Anforderungskataloges (z. B. in der Form eines

Pflichtenheftes) werden in einem Softwareentwicklungsprojekt fachliche Konzepte erarbeitet. Bei der Erarbeitung dieser Konzepte sollten die Anwender den fachlichen Input liefern, da man davon ausgehen kann, dass sich Entwickler nicht mit fachlichen Gegebenheiten, z.B. mit Feinheiten der Steuergesetzgebung, der Gebäudestatik, den Geschäftsprozessen an den Börsen oder den fachlichen Gegebenheiten eines Pharmaunternehmens auskennen. Entwickler müssen ihrerseits aber die Methoden und Techniken eines Fachkonzeptes beherrschen, um die zuvor formulierten Anforderungen weiter zu präzisieren. Hierbei geht es insbesondere um die exakte Beschreibung der erforderlichen Datenbasis, der Beschreibung der notwendigen Prozesse und der fachlichen Funktionen. Bei einem Fachkonzept muss man zu präzisen – und nicht beliebig interpretierbaren – Festlegungen kommen, die dann die Grundlagen für die IT - technische Spezifikation des zukünftigen Systems bilden. Ein Fachkonzept wird hierbei insbesondere mit grafischen Modellen (z. B. auf der Grundlage des derzeitigen Industriestandards UML = Unified Modelling Language) erstellt. Die Vorgaben für diese Modelle sind vom Anwender zu liefern, der Entwickler/Analytiker hat sie zu modellieren und der Anwender hat danach die Aufgabe, die erstellten Modelle nachzuvollziehen und ihre fachliche Korrektheit zu bestätigen oder aber zu verbessern. Analog kann man sich diesen Prozess so vorstellen, dass ein Bauherr nach Auftragserteilung auf der Grundlage eines Leistungsverzeichnisses (Pflichtenheft) mit dem Architekten die Aufteilung der Räumlichkeiten in einer Etage bespricht, der Architekt zeichnet nach den Angaben des Bauherrn einen Plan und der Bauherr sollte den Plan entweder in der vorliegenden Form akzeptieren oder aber z. B. bemängeln, dass die Kinderzimmer doch beängstigend klein geraten sind.

Solche Modelle sind nicht nur ‚Bildchen', sondern handfeste Festlegungen. Zudem sind für die Erstellung eines grafischen Modells sehr viele Fragen an den Anwender zu richten. Jede Beantwortung einer Frage klärt einen spezifischen Zusammenhang und ist somit eine Risikominderung für die gesamte Entwicklung. Zur Verdeutlichung dieses

Zusammenhangs sei ein kleines Beispiel eines sog. Klassenmodells nach UML aufgeführt.



Mit einem Klassenmodell wird u. a. dargelegt, welche Objekte für die Anwendung relevant sind, welche ihrer Eigenschaften relevant sind und wie die Objekte zueinander in Beziehung stehen. Im vorliegenden Beispiel einer Bibliotheksanwendung müssen sich Anwender und Entwickler zunächst über die Begrifflichkeit einigen. Hier wären z.B. folgende Fragen zu klären: Was ist ein Leser? Muss ein Leser lesen können, um ein Leser im Sinne des Systems zu sein? Wie alt muss ein Leser mindestens sein? Danach ist festzulegen, welche Eigenschaften (Attribute) ein Leserobjekt besitzen soll. Ebenso ist gemeinsam zu klären, was man unter einem Buch versteht. Hierbei wird sich sehr schnell erweisen, dass dieser Begriff nur sehr schwierig abgrenzbar ist. So kann es von einem Buch mehrere Exemplare in einer Bibliothek geben. Besitzt man dann auch mehrere Bücher? Weiterhin kann ein Buch in mehreren Bänden erscheinen. Ist ein Band ein Teil eines Buches oder auch ein Buch? Es können auch am Jahresende Zeitschriften gebunden werden. Besitzt man dann Bücher oder immer noch Zeitschriften? Vielleicht wäre es unter dem Aspekt der Katalogisierung und der Ausleihe dann geschickter, zunächst nur allgemein von bibliothekarischen Einheiten zu sprechen? Im Beispiel ist zudem modelliert, dass ein Leser kein, ein oder aber mehrere Bücher ausleihen kann (dargestellt durch 0..*). Hier wäre durch den Entwickler die Frage zu stellen, ob es bei der Zahl der Ausleihungen durch einen Leser eine Obergrenze gibt oder nicht? Ebenso ist im Modell vermerkt, das ein Buch an keinen oder an 1 Leser (dargestellt durch 0..1) zu einem Zeitpunkt ausgeliehen sein kann. Damit ist auch explizit festgelegt, dass im zukünftigen System keine Historie (Welche Leser hatten in der Vergangenheit das Buch ausgeliehen?) geführt wird. Würde der Anwender eine andere Vorgehensweise wünschen, müsste das obige Modell angepasst werden. Man kann sich sicher bereits bei diesem kleinen Beispiel vorstellen, wie eine Systementwicklung verlaufen

würde, wenn man all die zuvor gestellten Fragen nicht gemeinsam klärt.
In ähnlicher Form wie zuvor geschildert, sind auch Prozesse zu modellieren, fachliche Funktionen zu erläutern und man muss sich darüber abstimmen, bei welchen Ereignissen (z.B. eine Schadensmeldung bei einer Versicherung) welche Prozesse und welche Funktionen in welcher Reihung auszuführen sind. Es bleibt somit festzuhalten, dass Fachkonzepte mit unterschiedlichen Modellen gemeinsam von Anwendern/Entwicklern aufzustellen sind. Auch Anwender müssen die Bereitschaft zeigen, die erstellten Modelle zu lesen und auch zu verstehen (in ähnlicher Form in der ein Bauherr auch den Plan eines Architekten lesen und verstehen muss, um sicher zu gehen, dass seine Vorstellungen auch korrekt umgesetzt worden sind). Sofern die Anwender nicht über das erforderliche Know-how (z. B. Wie ist ein UML – Diagramm zu interpretieren?) verfügen, sollten entsprechende Schulungen/Einweisungen erfolgen, damit eine Kommunikation über die Modelle möglich ist.
Im Anschluss an das Fachkonzept ist das IT - System (oder aber bei agiler, iterativer Vorgehensweise Teile hiervon) zu spezifizieren, zu codieren und zu integrieren. Bei diesen IT – nahen Tätigkeiten ist üblicherweise keine Mitwirkung der Anwender erforderlich.

zu c) Mitwirkung bei der Gestaltung der grafischen Benutzeroberfläche (GUI) und von Benutzerhandbüchern

Die Anwender sollten im Rahmen eines Softwareentwicklungsprojektes darauf drängen, möglichst frühzeitig und intensiv in die Gestaltung der Benutzeroberfläche (Masken, Menues) eingebunden zu werden. Immerhin ist die Benutzeroberfläche der Teil der Software, der bei der späteren Nutzung für die Akzeptanz ausschlaggebend ist. Die Oberfläche sollte so gestaltet sein, dass fachliche Arbeitsabläufe unterstützt und erleichtert werden und dass es zu keiner Behinderung und Verzögerung kommt.
Weiterhin sollte sichergestellt werden, dass auch ein Benutzerhandbuch (bzw. –anleitung) so geschrieben wird, dass es für einen Anwender verständlich ist. Da Entwickler oftmals keine großen Literaten sind und zudem gewohnt sind, in einem „Fachchinesisch“ zu kommuni-

zieren, besteht die Gefahr, dass von Entwicklern verfasste Benutzerhandbücher für Anwender nicht verständlich sind. Somit ergibt sich auch in diesem Bereich eine Notwendigkeit zur Abstimmung und Zusammenarbeit.

d) Lieferung von fachlichen Testfällen, an Hand derer die fachliche Korrektheit eines Systems festgestellt werden kann und Abnahme des Softwaresystems.

Ein Anwender sollte bereit sein, Entwicklern Testfälle aus seiner fachlichen Sicht bereit zu stellen. Mit diesen fachlichen Testfällen kann dann die fachliche Korrektheit festgestellt werden. Die Testfälle sollten zwei Voraussetzungen erfüllen:

Die Testfälle sollten in dem Sinne überdeckend sein, dass von den zusammengestellten Testfällen alle Bereiche der Anwendung abgedeckt werden.
Die Testfälle sollten nicht nur den Normalfall abdecken, sondern auch komplexere fachliche Spezialfälle mit Grenzwerten umfassen, um Schwächen eines Systems aufzudecken.

Die Formulierung von fachlichen Testfällen durch Anwender ist sinnvoll, da sich diese – im Gegensatz zu Entwicklern - besser mit den fachlichen Gegebenheiten auskennen. Die Lieferung von Testfällen durch Anwender sollte aber nicht dazu verleiten, dass Entwickler aus ihrer Sicht keine weiteren Testfälle zusammenstellen. In der Praxis kommt es leider häufig vor, dass Anwendern der Test von Systemen überlassen wird. Für Entwickler ist dies eine komfortable Situation, da dann davon auszugehen ist, dass man das System nicht exzessiv und ausführlich testet. Fakt ist, dass die Mehrzahl der Entwickler nicht über vertiefte Testkenntnisse verfügt. Man kann sich dann leicht vorstellen, welches Test Know-how auf Anwenderseite vorhanden ist. Um es zu überspitzen, wäre der ausschließliche Test eines komplexen Softwaresystems durch Anwender mit der Situation vergleichbar, dass Entwickler von Formel - 1 Boliden ihren neuen Wagen ihrer Großmutter (oder –vater) zum Test überlassen. Richtig wäre es somit, wenn fachlich orientierte Testfälle von Anwendern bereitgestellt werden und zudem durch Testfälle ergänzt werden,

die durch Testspezialisten bzw. Entwickler geliefert werden. Letzteres ist unbedingt notwendig, da ein System auch dann noch Fehler enthalten kann, wenn alle fachlichen Funktionen korrekt ausgeführt werden. Um dies zu verstehen, muss man sich nur vergegenwärtigen, dass ein System nicht nur vom Anwender gewünschte Funktionen korrekt umsetzen kann, sondern auch unerwünschte – und natürlich auch nicht geforderte Funktionalitäten – ausführen kann. Zum Beispiel wird eine Versicherungsprämie korrekt berechnet und am Bildschirm angezeigt, zudem werden aber in fataler Weise die Daten des zugrunde liegenden Versicherungsvertrages von der Festplatte gelöscht oder – was noch schlimmer wäre, weil es schwerer aufzudecken ist - die Daten des Versicherungsvertrages werden nach der Prämienberechnung verfälscht abgespeichert.

Wenn ein System entwickelt und getestet worden ist, sollte ein Anwender sich die Ergebnisse der durchgeführten Test zeigen lassen und dann eine Entscheidung darüber herbeiführen, ob das System abgenommen werden kann (vielleicht trotz noch vorhandener Mängel, die noch auszubessern sind) oder wegen einer zu hohen Mängeldichte abgewiesen werden muss. Es ist hierbei explizit zwischen einer Abnahme als politischen Akt, der z. B. Zahlungen nach sich zieht und Gewährleistungsfristen festlegt und einem Test zu unterscheiden, dessen Ziel es sein sollte, Fehler aufzudecken.

Zusammenfassend lässt sich festhalten, dass eine komplexe Softwareentwicklung nur dann risikoarm und professionell durchgeführt werden kann, wenn Anwender und Entwickler intensiv miteinander kommunizieren und wenn jede Seit auch akzeptiert, dass es Aufgaben im Softwareentwicklungsprozess gibt, die in sinnvoller Weise nur gemeinsam bewältigt werden können. So gesehen, sitzt man in einem Boot, das nur dann den Fährnissen des Wassers trotzen kann, wenn man akzeptiert, dass man zur Zusammenarbeit verpflichtet ist.

Prof. Dr. Jürgen Spielmann

ist seit 1984 Professor für Softwaretechnik an der Fachhochschule Würzburg-Schweinfurt. Er verfügt über langjährige Beratungserfahrung und ist vereidigter Sachverständiger für Softwaretechnik und Softwarequalität.