MIND THE GAP: GAP-ANALYSEN ALS GEFAHR IN KOMPLEXEN SOFTWAREPROJEKTEN

Die Gap-Analyse ist eine in der Industrie weit verbreitete Methode zur Ermittlung von Anforderungen. Sie beschreibt die Diskrepanzen zwischen einer Referenzinstallation und dem vom Kunden gewünschten Verhalten einer Software. Dabei entfallen jedoch grundlegende Elemente der Anforderungsentwicklung – oft werden unvollständige oder falsche Anforderungen aufgenommen. Das Resultat sind hohe Folgekosten oder sogar ein Projektabbruch. Der Artikel zeigt, warum die Gap-Analyse gerade bei hoch komplexen Softwareprojekten keine probate Methode zur Anforderungsentwicklung ist.

die autoren

Dr. Siamak Tadjiky
(E-Mail: siamak.tadjiky@s4m.com),
Vice President der zum Bertelsmann Konzern gehörenden SM4 GmbH, ist dort für den Gesamtbereich Broadcast Management Systems verantwortlich.

Norman Dannhoff
(E-Mail: norman.dannhoff@s4m.com)
war verantwortlich für Module von Großanwendungen und betreut seit 2008 als technischer Projektkoordinator Großkunden bei S4M.

Komplexe Softwaresysteme, die eine Vielzahl individueller Workflows und spezifischer Anforderungen abdecken sollen, gibt es naturgemäß nicht von der Stange. Die entsprechenden Projekte sind aufwändig und das daraus resultierende Projektvolumen macht sie für Softwarehersteller entsprechend interessant. Daher kommt es – sobald ein Unternehmen seinen Wunsch nach neuer maßgeschneiderter Software zur Abbildung eigener Geschäftsprozesse publik gemacht hat – immer wieder zu folgender Standardsituation.

Kunden gewinnen, Kosten kleinhalten

Die Bewerber um das Projekt stehen Schlange und im harten Kampf um den Auftrag versuchen die verschiedenen Bieterparteien, ihre Angebote möglichst überzeugend zu präsentieren. Die bekannte hohe Anzahl gescheiterter IT-Projekte lässt jedoch so manchen Auftraggeber vorsichtig agieren. So spielen bei der Auftragsvergabe längst nicht nur Kosten, Lieferumfang und prognostizierte zeitliche Projektlänge eine Rolle. Vielmehr möchte der Auftraggeber überzeugt sein, dass der Auftragnehmer auch in der Lage ist, das gewünschte Projekt erfolgreich zu realisieren. Diese erfolgreiche Realisierung eines IT-Projekts ergibt sich nicht nur aus der finalen Auslieferung einer Software, sondern vor allem aus dem Grad der Erfüllung der Ziele, die der Auftraggeber an die neue Software geknüpft hat.

Um den potenziellen Kunden davon zu überzeugen, den speziellen Anforderungen gewachsen zu sein und die gewünschten Ziele erreichen zu können, präsentieren die Bewerber im Normalfall bereits abgeschlossene Referenzprojekte aus derselben Produktlinie. Überzeugt ein solches Referenzprodukt den Kunden, ist ein großer Schritt zum Vertragsabschluss getan – die Anforderungen an das neue System können ermittelt werden, um neben den Kosten für die neue Software auch den Lieferumfang und den Lieferzeitpunkt abzuschätzen.

Doch die Entwicklung von Anforderungen für eine Software ist eine ebenso teure wie existenzielle Disziplin. Hier offenbart sich nun ein großer Konflikt: Die Qualität der Anforderungsentwicklung ist in hohem Maße sowohl von der investierten Zeit als auch von der Qualität der Anforderungsentwickler abhängig. Ein qualifizierter Anforderungsentwickler muss verschiedene Techniken zum Erheben von Anforderungen beherrschen. Er muss in der Lage sein, diese Anforderungen zu analysieren und gegebenenfalls zu modellieren. Eine gewaltige Aufgabe, die nicht nur eine gute Ausbildung verlangt, sondern auch viel Erfahrung. Und all das sind große Kostenfaktoren.

Aus diesem Grunde ist neben der vollständigen Form der Anforderungsentwicklung (Erheben, Analysieren, Spezifizieren, Prüfen, vgl. [Res]) eine parallele Disziplin entstanden: die Gap-Analyse.

Merkmale der Gap-Analyse

Bei der Gap-Analyse wird der Unterschied zwischen einer Anforderung und einem bereits existierenden System beschrieben. Besonders relevant sind Gap-Analysen daher in Wartungsprojekten und bei der Wiederverwendung von Software. Sie bergen jedoch immer die Gefahr, dass Eingriffe in ein bestehendes Softwaresystem unterschätzt werden und dass zu spät erkannt wird, dass sie umfassende Seiteneffekte haben, die dann nur noch mit Mehraufwand beherrschbar sind. Wichtig für Gap-Analysen sind die Qualität der bestehenden Dokumentation und der Umfang der Änderungen (vgl. [Ebe08]).

Zunächst ergeben sich Kosteneinsparungen, indem die drei Prozessschritte *Erheben, Analysieren* und *Prüfen* aus den bestehenden Dokumentationen übernommen werden: Weder wird das notwendigerweise hoch qualifizierte Personal zur Ausführung dieser Tätigkeiten benötigt, noch muss die zur Abarbeitung der ent-

sprechenden Aufgaben erforderliche Zeit bereitgestellt werden. Und auch der Kunde fühlt sich oftmals bei dieser Art der Anforderungsentwicklung sehr wohl, denn bereits zu diesem sehr frühen Zeitpunkt ist ein reales Produkt vorhanden, das viel Sicherheit und Orientierung vermittelt.

Schwächen der Gap-Analyse

Die Gap-Analyse scheint also zu diesem Zeitpunkt für beide beteiligte Parteien Vorteile zu bieten. Doch welche Folgen hat das beschriebene Vorgehen für den weiteren Projektverlauf (siehe Abbildung 1)?

Nur „ähnlich" reicht nicht

Wenn man sich am bereits fertigen Produkt orientiert, ist die Sicht auf das gewünschte Produkt nicht mehr wertfrei. Der gesamte Entwurf des Neusystems ist damit getrieben vom Erscheinungsbild des Referenzprodukts und auf dessen Paradigma ausgerichtet. Der Versuch, ein neues Phänomen durch ein bestehendes Framework abzudecken, wird mit großer Hartnäckigkeit weiterverfolgt, denn der Erfolg einer bestehenden Lösung verbreitet die Zuversicht, dass eine Lösung existiert – auch wenn sie schwer zu finden ist (vgl. [Kuh62]). In der Folge werden existierende Geschäftsprozesse oder -vorfälle nicht mehr korrekt festgehalten, sondern zunächst in das Korsett des Referenzprodukts gezwängt. Auf den ersten Blick wirkt dies oft elegant, denn es scheint so, als könne ein Geschäftsvorfall durch eine bestehende Funktionalität abgedeckt werden. Erst bei der technischen Analyse durch die Entwicklung oder zu einem noch späteren Zeitpunkt wird festgestellt, dass Geschäftsvorfall und eine bestehende Funktionalität zwar Ähnlichkeiten aufweisen, jedoch nicht kompatibel sind. Damit ist nicht nur die an diesen Geschäftsvorfall geknüpfte Aufwandsabschätzung falsch, sondern ein gesamtes existierendes technisches Konzept kann hinfällig sein. Zeit- und Kostenplan geraten unter Druck oder können nicht erfüllt werden.

Projekt fertig – Ziele verfehlt

Die Gap-Analyse vernachlässigt die Ziele, die mit einem System erreicht werden sollen. Das Eruieren von Zielen ist eine der wichtigsten Aufgaben der klassischen Anforderungsentwicklung, denn diese Ziele fokussieren die Problemdomäne und die tatsächlichen Bedürfnisse der Stakeholder. Die Anforderungsanalyse beginnt mit der Systemidee und Zielsetzung, in der geklärt wird, was das eigentliche Ziel der Entwicklung sein soll (vgl. [Oes09]). So ist es beispielsweise durchaus möglich, dass zwei Unternehmen in ähnlichen oder sogar in denselben Geschäftsfeldern arbeiten, ihre Ziele jedoch grundlegend andere sind. Beispielsweise kann Unternehmen *A* ein rein kommerzielles Unternehmen sein, das einen möglichst großen Profit anstrebt, während Unternehmen *B* ein staatliches Unternehmen ist, dessen Ziel es ist, Arbeitsplätze zu erhalten oder kulturelle Werte zu vermitteln. Sollte also das für Unternehmen *A* entwickelte Produkt für Unternehmen *B* als Referenz dienen, so ist es durchaus möglich, dass Unternehmen *B* seine Geschäftsprozesse abarbeiten könnte. Die Ziele, die Unternehmen *B* verfolgt – wie z. B. sozialer Frieden im Unternehmen oder hohe kulturelle Qualität – werden jedoch nicht erreicht. Das Projekt wäre zunächst gescheitert oder müsste aufwändig nachbearbeitet werden.

Zerstörung der Struktur

Die Gap-Analyse konzentriert sich auf Probleme, die in der Lösungsdomäne lokalisiert werden. Dies impliziert, dass auch die Lösungen dieser Probleme in der Lösungsdomäne angesiedelt sind. Mit anderen Worten schreibt die Gap-Analyse einer Entwicklungsabteilung bereits vor, wie eine technische Lösung zu implementieren ist. Dieses Vorgehen kann gravierende Folgen haben. Zunächst besteht die Möglichkeit, dass die beschriebene technische Lösung nicht umsetzbar ist. Das Projekt wäre gescheitert. Weiterhin besteht die Möglichkeit, dass beschriebene technische Anforderungen sehr aufwändig sind und dass das Projekt aus dem Zeit- und Kostenplan läuft.

Auch der Auftragnehmer kann schwer in Mitleidenschaft gezogen werden, wenn technische Lösungen vorgeschrieben werden. In der Regel verfügen Unternehmen, die Software produzieren, über eigene Frameworks und Architekturen, die die Funktionsweise des fertigen Softwareprodukts über definierte Zuständigkeiten, Komponenten und Schnittstellen sicherstellen. Diese Art der Komposition von Software bietet dem Unternehmen große wirtschaftliche Vorteile. Werden nun jedoch Anforderungen festgeschrieben, die sich mit der Architektur nicht vereinbaren lassen, führt das zum Konflikt. Häufig wird in diesen Situationen aus Zeitmangel gegen eine intakte Architektur entschieden (wenn überhaupt noch entschieden wird) und ein die Architektur verletzender Code wird implementiert. Das führt jedoch zu einem stetigen Verfall der Architektur (der in der Literatur als *Architectural Decay* beschrieben wird, vgl. [May99]) und damit zum Verfall der

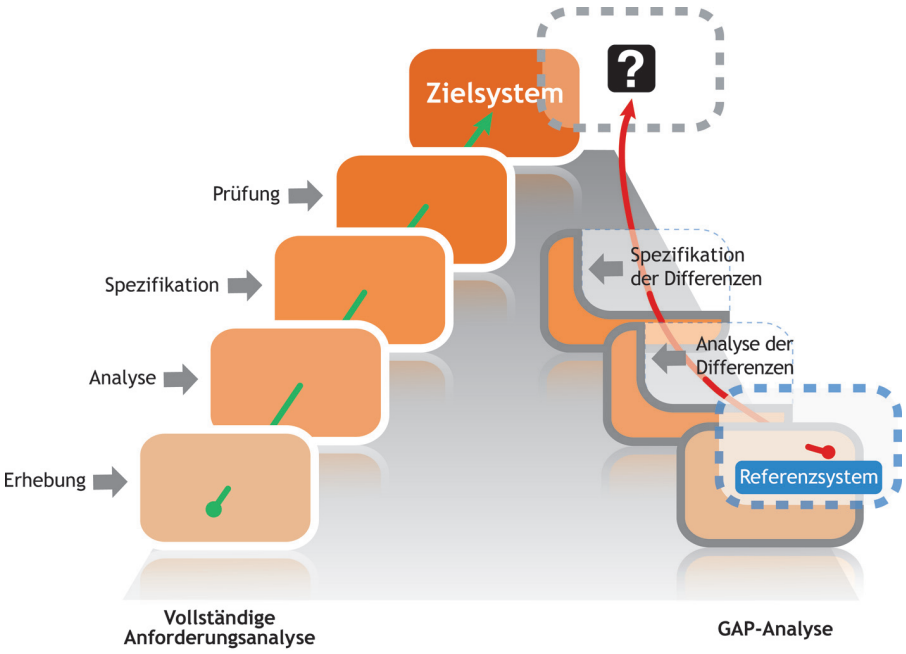

Abbildung 1: Projekt fertig, Ziel verfehlt.

Bereich	Soll (Vorteile der Methode)	Ist (Nachteile der Methode)
Ziele	Die Ziele des Wunschsystems sollen erfüllt werden.	Die Ziele des Referenzsystems werden erfüllt.
Geschäftsvorfälle	Vollständig beschriebene Anforderungen, und damit optimale technische Lösungen.	Erzwungene technische Lösungen, die nicht optimal sind und die mit den Geschäftsprozessen des Referenzsystems kollidieren.
Prozessbewertung	Die Qualität der Anforderungen und des Anforderungsprozesses kann gemessen und verbessert werden.	Schwächen des Anforderungsprozesses sind nicht sichtbar bzw. werden falsch zurückgekoppelt.
Änderungsmanagement	Eindeutig und klar definierte Anforderungen, die es zulassen, Abweichungen als Änderungsanforderungen zu erkennen.	Unscharfe Anforderungen, die es nicht zulassen, zwischen falschen/unvollständigen Implementierungen und Änderungsanforderungen zu unterscheiden.
Kosten	Geringerer zeitlicher Aufwand bei der Anforderungserhebung bedeutet einen positiven Kosteneffekt.	Folgekosten durch erforderliche Nachbesserungen sind überproportional höher als der ursprüngliche Mehraufwand.

Tabelle 1: Vermeintliche Vorteile und das tatsächliche Ergebnis.

wirtschaftlichen Vorteile. Das Ergebnis: Das aktuelle Projekt ist gerettet, die Folgeprojekte sind jedoch bereits angezählt. Unterschiedliche Beispiele haben gezeigt, dass die Entwicklung neuer Produkte nicht möglich war, weil die vorhandene Software die dazu erforderlichen Änderungen nicht mehr erlaubte (vgl. [Raj00]).

Gap-Analysen fördern Unschärfen

Scheitern Teile eines Projekts oder sogar ein gesamtes Projekt oder läuft ein Projekt aus dem Kostenplan, hat das weitreichende Folgen. Einerseits müssen die problematischen Prozesse in der Anforderungs- und Softwareentwicklung analysiert und verbessert werden, andererseits müssen die verantwortliche Partei und der Träger des entstandenen Schadens bestimmt werden.

Das jedoch ist problematisch: Die durch die Gap-Analyse entstandene Unschärfe in den Anforderungen zieht sich nun wie ein roter Faden durch das gesamte Projekt und verursacht mit dem Fortschreiten des Projekts einen ständig wachsenden Schaden. Die zum Zeitpunkt der Anforderungsentwicklung getroffenen falschen Annahmen und Interpretationen über eine bereits existierende Funktionalität, Missverständnisse und Oberflächlichkeit haben sich prozessübergreifend ausgedehnt und zu falschem Programmcode oder sogar zu einer fehlerhaften Architektur geführt. Damit dehnt sich auch der Konflikt weiter aus, denn nun zanken nicht nur Kunde und Anforderungsentwicklung über falsche Software, auch die Anforderungs- und die Softwareentwicklung streiten über falsche Implementierungen. So entstehen die bekannten weiteren Kosten (vgl. [Boe81]). Besonders problematisch ist hierbei, dass sich diese Kosten kaum noch zuordnen lassen, denn Neuanforderungen und Fehlermeldungen sind nicht mehr zu unterscheiden.

Warum dann Gap-Analysen?

Die oben erläuterten Schwächen der Gap-Analyse verdeutlichen die große Problematik dieser Vorgehensweise (siehe Abbildung 2). Je mehr Kunden ein Unternehmen in eine Software-Produktlinie integriert, desto dramatischer werden die Folgen. Aber warum werden dann weiterhin Gap-Analysen betrieben? Die Gründe dafür sind vielfältig und variieren in ihrer Gewichtung von Projekt zu Projekt oder Unternehmen zu Unternehmen. In ihrem Charakter sind sie jedoch sehr ähnlich.

Interne Abläufe

Die Anforderungs- und die Softwareentwicklung werden zwar häufig vom Softwareunternehmen selbst durchgeführt, beide Disziplinen sind jedoch größtenteils in unterschiedlichen Unternehmensbereichen angesiedelt. Nach Beendigung der Anforderungsentwicklung werden deren Ergebnisse der Software-Entwicklungsabteilung zugeführt, die dann versucht, diese umzusetzen. Schlussendlich wird ein Softwareprodukt an den Kunden ausgeliefert.

Wurden die Anforderungen nicht korrekt aufgenommen, entspricht die entwickelte Software daher nicht den Ansprüchen des Kunden. Dieser kann demzufolge seine an die Software geknüpften Ziele nicht erreichen. Eine Nachbearbeitung des Systems ist notwendig und der Termindruck entsprechend hoch. Der möglichst schnellen Umsetzung wegen werden die gewünschten Änderungen an der Software – unter Umgehung der Anforderungsentwicklung – pragmatisch direkt an die Softwareentwicklung weitergegeben. Und genau dort entstehen nun auch die Kosten. Das bedeutet, dass Fehler, die in einem Unternehmensbereich (Anforderungsentwicklung) begangen werden, von einem anderen Bereich (Softwareentwicklung) sowohl korrigiert als auch finanziell getragen werden.

Die Fehler der Anforderungsentwicklung haben also wenig oder keinen Einfluss auf den Anforderungsentwicklungsprozess, denn sie werden in diesem Bereich weder kommuniziert noch korrigiert. So bleiben die großen Schwächen der Gap-Analyse verborgen. Da der auslösende Bereich keine negativen Rückkopplungen erhält, gibt es vordergründig keine Notwendigkeit für Veränderungen.

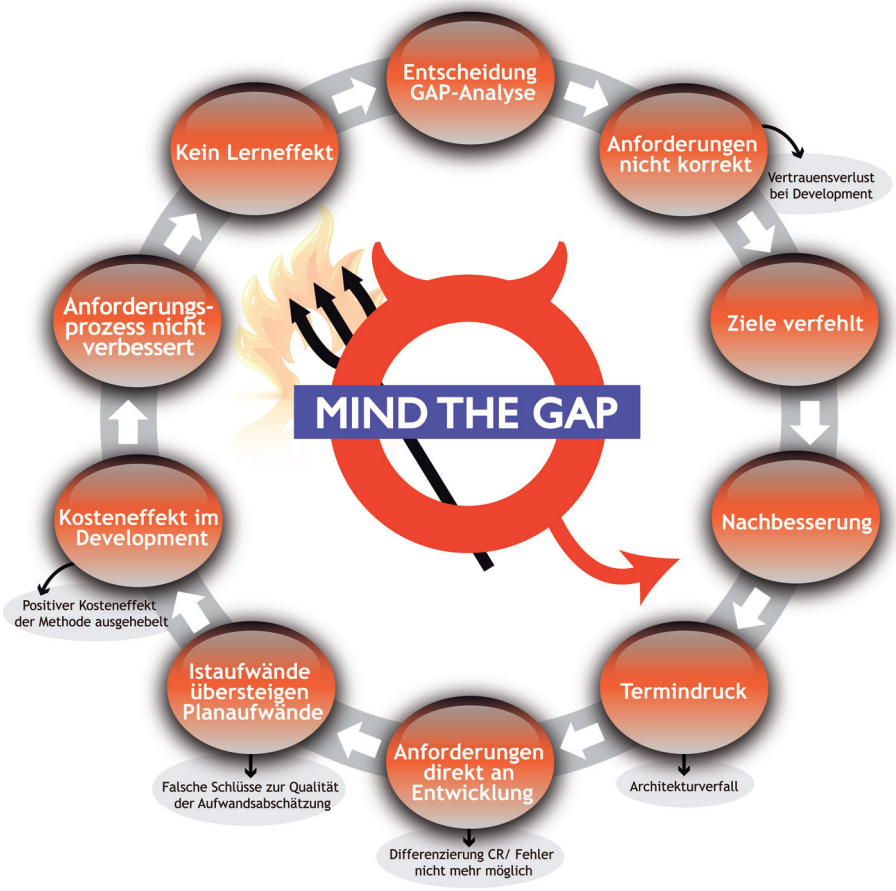

Abbildung 2: Der Teufelskreis bei GAP-Analysen.

Schlechte Erfahrungen

Eine Produktentwicklung setzt zwar eine Anforderungsentwicklung voraus, die Wichtigkeit der vollständigen Anforderungsentwicklung wird jedoch oftmals unterschätzt.

Dieses Verhalten begründet sich in einem erlernten Reiz-Reaktions-Muster von Projektteilnehmern. Eine ehemals erlebte falsche oder unzureichende Anforderungsentwicklung führt zu der Schlussfolgerung, dass die Anforderungsentwicklung als solche unbrauchbar ist (vgl. [Hec80]). Dokumente, die mühsam erstellt wurden, jedoch die Softwareentwicklung wenig oder nicht unterstützen oder sogar behindern, sorgen für einen Vertrauensverlust der gesamten Disziplin der Anforderungsentwicklung auch in folgenden Projekten und ebnen den Weg für scheinbare Alternativen – wie die Gap-Analyse.

Fehlendes Wissen

Häufig werden die korrekten Ergebnistypen der vollständigen Anforderungsentwicklung von Softwareentwicklern entweder nicht erkannt oder nicht ernsthaft eingesetzt.

Ist der Entwickler oder Architekt aus mangelnder Methodenkompetenz nicht in der Lage, die Ergebnistypen der Anforderungsentwicklung korrekt zu interpretieren und einzusetzen, verfallen diese und werden als nutzlose, teure Beilagen empfunden. Die vollständige Anforderungsentwicklung erscheint als Prozess, der viel Kosten verursacht, jedoch wenig verwertbaren Output liefert. Eine Gap-Analyse gewinnt dadurch in großem Maße an Attraktivität: Sie liefert verführerisch schnell technisch verwertbare Informationen – allerdings sind diese oft unvollständig, oberflächlich oder falsch. Software-Entwicklungsabteilungen müssen also lernen, mit den abstrakteren Ergebnissen der vollständigen Anforderungsentwicklung umzugehen und diese zu nutzen.

Zusammenfassung

Im Allgemeinen lässt sich sagen, dass Änderungen unterhalb von 10 % des Gesamtsystems mit Gap-Analysen beherrschbar sind. Änderungen oberhalb von 10 % bis ungefähr 30 % müssen hinsichtlich ihrer Seiteneffekte bereits sehr gut analysiert werden.

Änderungen, die mehr als 30 % des bestehenden Umfangs umfassen, oder auch nicht-funktionale Anforderungen, die im Nachhinein umgesetzt werden sollen, erfordern in aller Regel eine Neuentwicklung der Anforderungen (vgl. [Ebe08]). Gap-Analysen führen in derartigen Szenarien zu großen Unschärfen innerhalb der Anforderungen und diktieren technische Lösungen, die oft nicht zu verwirklichen sind. Ein geregelter Projektablauf kann damit nicht mehr gewährleistet werden: „Ex falso sequitur quodlibet" (aus Falschem folgt Beliebiges). ■

Literatur & Links

[Boe81] B.W. Boehm, Software Engineering Economics, Prentice-Hall 1981

[Ebe08] C. Ebert, Systematisches Requirements Engineering und Management, dpunkt.verlag 2008

[Hec80] J. und H. Heckhausen Motivation und Handeln, Lehrbuch der Motivationspsychologie, Springer 1980

[Kuh62] T.S. Kuhn, The Structure of Scientific Revolutions, University of Chicago Press 1962

[May99] A. v. Mayrhauser, J. Wang, Fault Architecture as an Indicator of Architectural Problems. In: Proc. Of Int. Conf. on Software Engineering (ICSE '99), Los Angeles 1999

[Oes09] B. Oestereich, Analyse und Design mit UML 2.3, Oldenbourg Verlag 2009

[Raj00] V.T. Rajlich, K.H. Bennet, A Staged Model for the Software Life Cycle, in: Computer, Juli 2000

[Res] re-wissen, Homepage, siehe: www.re-wissen.de