
Software Development - Vom Business Case zum Produkt

Version 1.0 - 22.02.2007, Claus Ehrenberg

aubergemediale

aubergemediale White Paper Series
www.aubergemediale.com

Software-Development - vom Business Case zum Produkt

Version 1.0 - 22.02.2007, Claus Ehrenberg

Inhalt

Business Planning ist nicht Produktspezifikation	3
Software Requirements Engineering	6
Vorteile durch gutes Software Requirements Engineering	7
Literatur	11

aubergemediale

Business Planning ist nicht Produktspezifikation

Am Anfang eine kleine Geschichte. Die Initiatoren der aubergemediale AG bewarben sich im Jahr 2005 erfolgreich um einen Projektzuschuss des Bundesministeriums für Forschung und Wissenschaft in größerer Höhe für ein Software-Entwicklungsvorhaben.

Die Förderkriterien des BmBF enthielten dabei zwei Hauptforderungen: Das Vorhaben mußte *Hochtechnologie* (hier: Software) betreffen und „Aussicht auf wirtschaftlichen Erfolg“ haben, der durch einen im Projektverlauf zu erstellenden Business Plan dokumentiert werden sollte.

Im besten Bewußtsein dieser beiden Forderungen (Hochtechnologie, Geschäftserfolg/Business Plan) wurde die vorhandene Idee, eine Social Web-Internetanwendung, dementsprechend ausgestaltet, indem versucht wurde, möglichst komplexe Konzepte der Informatik zur Realisation zu verwenden (Anforderung Hochtechnologie). Um den Geschäftserfolg im Business Plan möglichst sicher erscheinen zu lassen, wurde erheblicher Aufwand in einem zeitweise siebenköpfigen Team betrieben, eine Vielzahl von Features zu entwickeln und sinnhaft zu vernetzen, um die Einnahmenseite des Business Plans möglichst gut aussehen zu lassen. Denn schließlich lautet die beliebteste Frage der Jury bei Business-Plänen zu E-Commerce-Projekten bekanntlich: *Und wie verdienen Sie Geld?*

Die Initiatoren waren hier durchaus sehr erfolgreich und entwarfen ein Konzept für ein sehr komplexes Produkt mit 20-30 möglichen Erlösquellen und einer Vielzahl attraktiver Features um die Akzeptanz beim Enduser zu sichern. Das entstandene Konzept manifestierte sich schließlich in

einem detaillierten und sogar prämierten Business Plan.

Die Probleme entstanden jedoch, als mit der Umsetzung begonnen werden sollte. Die Programmierung des beschriebenen Software-Systems geriet nach anfänglichen Fortschritten immer mehr ins Stocken bis das Projekt schließlich auf Eis gelegt wurde. Was war also passiert?

Typisch für Business Pläne ist, dass vor allem ökonomische Aspekte ausgeführt werden. Die Geschäftsidee, das Team, die Technologie, Markt und Wettbewerb, Chancen und Risiken werden kritisch betrachtet. Das eigentliche Produkt wird in unterschiedlichster Weise *diskutiert*, aber es wird nicht *spezifiziert*. Kurz gesagt: Baupläne gehören normalerweise nicht in den Business Plan, schon weil sich dieses Dokument an Betriebswirtschaftler und nicht an Ingenieure richtet.

Aber welche Relevanz hat dieser Umstand nun? Das Problem ist, dass die im Business Plan enthaltenen Informationen noch keine ausreichende Grundlage bilden, ein Software-Projekt schnell umzusetzen. Trotz evtl. enthaltenen *Milestones*, *nächsten Schritten* Zeit- und Aufwandsschätzungen fehlen andere wichtige Elemente wie Projektpläne oder eben die „*Baupläne*“ der Software.

Produktbeschreibungen in Romanform arbeiten mit der Phantasie des Lesers. Gerade auch in Business Plänen möchte man den Leser nicht langweilen sondern auch für das Projekt begeistern. Es herrscht ein Zwang zur Beschränkung auf das *Wesentliche* (um z.B. einen möglichen Investor zu überzeugen). Auch wird viel mehr auf den Zweck bzw. Nutzen des Produktes abgestellt als auf die konkrete Ausgestaltung. Ein Business Plan ist schlicht und ergreifend vollkommen unzulänglich, was die Spezifikation des Produktes betrifft. Man kann ein Produkt nicht anhand eines Business Plans implementieren, es fehlen

noch einige Zwischenschritte. Daraus folgt ebenfalls, dass man die Herstellungskosten eines Produktes nicht anhand des Business Plans bestimmen kann, wenn es sich um ein Entwicklungsvorhaben handelt. Damit entsteht hier eine gefährliche Lücke, die oft nicht auf den ersten Blick wahrgenommen wird.

Ein ähnlicher Aspekt ist der, dass der Business Planer nicht die Möglichkeit hat, einfach ein Angebot zur Ermittlung der Herstellungskosten einzuholen. Ein Anbieter von Programmier-Dienstleistungen benötigt zur Abschätzung der Entwicklungskosten eine vollständige und valide Aufstellung aller benötigten Features und Anforderungen. Diese könnte aus dem Business Plan jedoch nur durch einen sehr hohen Anteil an purer Spekulation abgeleitet werden. Davon abgesehen liegt diese Transformationsleistung auch nicht im Kompetenzraum von Programmierern, sondern erfordert eher Qualifikationen des Marketings oder eben entsprechende Spezialisten (Business Analyst, Anforderungsanalytiker).

Außerdem beschreiben Business Pläne aus ihren oben erwähnten Zielsetzungen heraus i.d.R. Maximallösungen. Eine maximale Zahl von Features soll den Kundennutzen und Markterfolg gewährleisten. Alle möglichen Erlösquellen sollen ausgeschöpft werden. Mangels detaillierter Spezifikation der Anforderungen wird nicht erkannt, dass jedes Feature zusätzlichen Aufwand verursacht bzw. der Aufwand wird gering geschätzt, nach dem Motto: wenn das Feature nicht viel bringt, dann kann es auch nicht viel kosten. Leider ist es aber nicht so. Die Kosten eines Features sind natürlich von ihrem Nutzen völlig unabhängig. Darüber hinaus haben alle Features in einem Produkt Querverbindungen und Abhängigkeiten. Daraus folgt, dass der Entwicklungsaufwand mit der Zahl der Features nicht linear sondern exponentiell steigt. Features

müssen daher bewertet und priorisiert werden. Auch das setzt selbstverständlich ihre Dokumentation in geeigneter Form voraus.

Wünschenswert wäre es auch, wenn bezüglich der Herstellung eines (Software-) Produkts eine fundierte *Make-Or-Buy*-Entscheidung getroffen werden könnte. Dabei ist es für die zügige Einholung von Angeboten von Zulieferern unabdingbar, das herzustellende Produkt gut, eindeutig und gegenüber verschiedenen potentiellen Zulieferern einheitlich zu beschreiben.

Software Requirements Engineering

Während es etwa bei Bauvorhaben oder zu entwickelnden Maschinen beinahe jedem klar ist, dass eine genaue Definition des zu entwickelnden Produktes nötig ist, wird diese basale Notwendigkeit von Managern oft übersehen, wenn es um die Software-Entwicklung geht. Zu erwarten, dass diese Aufgabe von den Programmierern erledigt wird, ist ungefähr das gleiche wie zu erwarten, dass die Monteure das Auto konstruieren oder die Maurer das Haus entwerfen. Verstärkt wird dieses Missverständnis vor allem dadurch, dass diese Aufgaben bei kleinen Software-Projekten tatsächlich regelmäßig von Programmierern übernommen werden müssen. Aber schon bei kleineren Projekten sind die negativen Folgen klar erkennbar und größere Software-Projekte sind notorisch bekannt für Zeit- und Kostenüberschreitungen, unzufriedene Kunden und Arbeitsüberlastung bei allen Beteiligten und eine hohe Zahl vollkommen abgebrochener Projekte, die in keiner anderen Branche zu einem so hohen Anteil zu finden sind. Belege für die fatalen Folgen eines fehlenden Requirements Engineering sind in der Literatur zahlreich zu finden, allerdings lesen Manager nur in den seltensten Fällen Fachliteratur über Software-Entwicklung und

das Thema ist zu trocken für die populären Wirtschaftsmagazine.

Die Frage, ob Software Requirements Engineering eine überflüssige Übung in übertriebener Sorgfalt oder schon ein *Overengineering* aus sich selbst heraus ist, kann klar mit „nein“ beantwortet werden. Selbst kleine Projekte profitieren davon. Der Aufwand für das Requirements Engineering ist proportional zum Gesamtaufwand in Zeit und Kosten und beträgt zwischen 10 und 20% des Gesamtaufwands. Der Schlüssel zu Erfolg liegt in einem angemessenen Verhältnis von Planung und Ausführung, nicht im Weglassen der Planung.

Vorteile durch gutes Software Requirements Engineering

Die Bestimmung der Anforderungen ist Produktgestaltung bis ins Detail und kein fertiges Produkt kann entstehen, ohne dass es vollständig gestaltet wird. Die Frage ist also nicht, ob ein Requirements Engineering durchgeführt wird, sondern auf welche Weise. Und der richtige Weg besteht keinesfalls darin, die Programmierer mit vagen Anweisungen drauflos arbeiten zu lassen und ab und zu korrigierend einzugreifen. Ergebnisse der Programmierung werden erst sichtbar, wenn sie bis ins Detail ausgeführt sind. Ein Feature, das ich mit wenigen normalsprachlichen Sätzen oder einer Grafik treffend beschreiben lässt benötigt zur Codierung in einer Programmiersprache einen ungleich höheren Aufwand. Und die normalsprachliche Definition würde keineswegs entfallen - bei fehlenden Spezifikationen muss sie statt dessen vom Programmierer „on the fly“ mit erledigt werden - sonst wüsste er ja nicht, was er überhaupt programmiert. Außerdem ist das „Auscodieren“ eines Programms eine Tätigkeit die Festlegungen in detail erfordert, während

gutes Produktdesign zunächst den Rahmen festlegt und sich zunächst mit den wichtigsten Aspekten auseinandersetzt. *Gutes Requirements Engineering bringt die nötigen Schritte in die richtige Reihenfolge und dadurch nur Vorteile.*

Die richtige Vorgehensweise liegt in der effektiven und effizienten Erreichung der Ziele. Wenn der Ausgangspunkt der Business Plan ist, bedeutet das vor allem:

- das Produkt muss in seiner Gestalt und Qualität genau die für den Markterfolg nötigen Anforderungen erfüllen (Effektivität).
- das Produkt muss möglichst kostengünstig, schnell oder im Rahmen des Budgets hergestellt werden.

Gutes Requirements Engineering dient genau diesen Zielen. Es ist auch vollkommen einleuchtend, dass das bestimmen der Produktgestalt und -Qualität noch wenig mit Software-Entwicklung zu tun hat. Die Geschäftsanforderungen, Endbenutzeranforderungen und Qualitätsmerkmale leiten sich zunächst aus den Business-Zielen, Marktuntersuchungen und sonstigen Bedingungen (z.B. rechtlich, organisatorisch, strategisch) ab. Das erste detaillierte Produktdesign findet also noch nicht in einer Programmiersprache, sondern in einer Sprache, die die maßgeblichen Personen verstehen müssen, statt. Aus der Perspektive des verantwortlichen Entscheiders würde ein Verzicht auf eine explizite und lesbare Produktspezifikation dem Verzicht auf wesentliche Möglichkeiten der Steuerung und Kontrolle gleich kommen.

Zweitens kann nur anhand einer genauen Produktspezifikation zielgerichtet und ohne Umwege programmiert werden. Qualitätsanforderungen, Benutzerzahlen und gewünschte Features müssen zunächst bestimmt werden, um

überhaupt die zu verwendende Technologie auszuwählen (z.B. Servertypen, Betriebssystem, Programmiersprache, Systemarchitektur). Im Extremfall können übersehene Anforderungen dazu führen, dass erst zu einem späten Zeitpunkt erkannt wird, dass sämtlicher schon unternommene Entwicklungsaufwand unbrauchbar ist, weil eine bestimmte Kundenanforderung sich auch mit aufwändigen Änderungen nicht erfüllen lässt, sondern eine Neuentwicklung erfordert.

Gleiches gilt für die Genauigkeit von Kostenschätzungen und Zeitaufwand. Diese können umso genauer oder überhaupt erst erfolgen, wenn die Spezifikation des Produktes ausreichend genau vorliegt. Die Vollständigkeit der Spezifikation ist dabei von großer Wichtigkeit. Schließlich können Features, die aus Sicht der Entscheider Kleinigkeiten zu sein scheinen, tatsächlich aber hoch aufwändig sein.

Und wie schon erwähnt, sind Änderungen umso teurer, je später im Entwicklungsprozess sie erfolgen. Es besteht nicht nur das Risiko, dass Arbeit weggeworfen werden muss. Vielmehr können Anforderungsänderungen auch Wirkungen entfalten, die sehr weitreichende Umbauten am System erfordern.

Im Ergebnis führt gutes Requirements Engineering damit genau zum gewünschten Ziel - dem besten Produkt zum besten Preis. Es liegt einfach in der Natur der Sache. Das Problem ist eher dass man fertiger guter Software diesen Aufwand nicht mehr ansieht und in verbreiteten naiven Vorstellung über das Programmieren begründet.

Der Aufwand für das Requirements Engineering wird jedoch belohnt. Das Ergebnis ist nicht nur die effektive und effiziente Zielerreichung (was ja schon genug wäre). Darüber hinaus führt es zu einer genaueren Beschäftigung

der verantwortlichen Personen mit dem Zielprodukt, was sich positiv auf Qualität und Markterfolg auswirkt.

Weiter entsteht ein großer zeitlicher Informationsvorsprung. Wenn das Produkt zunächst normalsprachlich definiert wird, kann besser parallel mit Marketingmaßnahmen begonnen werden. Detailreiche, gute Beschreibungen können bei nötigen Kapitalbeschaffungsmaßnahmen eine entscheidende Rolle spielen, weil durch sie klar wird, wie weit das Projekt schon gediehen ist und dass es professionell organisiert ist. Anhand von guten Beschreibungen können Vertriebsleute besser nach zusätzlichen Absatzmöglichkeiten für eventuelle abgeleitete Produkte suchen. Während ein Business Plan im Bereich Software heutzutage nur noch selten einen Wert für sich darstellt, ist eine Software Requirements-Spezifikation (SRS) schon ein werthaltender Aktivposten (sogar in steuerlicher/ bilanzieller Hinsicht, da die Aufwendungen aktivierbar sind).

Vollständige Requirements sind auch die Voraussetzung einer optimalen Projektorganisation. Anhand einer SRS kann die Gesamtaufgabe effektiv auf verschiedene Personen oder Dienstleister aufgeteilt werden. Die Kenntnis der Gesamtheit schafft erst den Überblick, welche Schritte aufeinander aufbauen oder parallel laufen können. Anhand von (Teil-) Spezifikationen ist es erst möglich, Angebote einzuholen und für beide Seiten funktionierende Verträge für Fremdleistungen, Outsourcing oder gar Offshoring zu schließen. ■

Literatur

Brooks, Frederick P., Jr., 1995. The Mystical Man Month: Essays on Software Engineering, Anniversary Edition (2d Ed), Reading, MA: Addison-Wesley

IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications.

IEEE Std 1062-1998. Recommended Practice for Software Acquisition.

McConnell, Steve, 2004. Code Complete, Second Edition. Redmond: Microsoft Press.

McConnell, Steve, 2006. Software Estimation. The Black Art Demystified. Redmond: Microsoft Press.

Wieggers, Karl E., 2005. Software Requirements, Second Edition. Redmond: Microsoft Press.

