Das Projekt fährt gegen die Wand, aber wer sitzt am Steuer?

Nico Zinner
Trivadis GmbH
Stuttgart

Schlüsselworte:

Softwareentwicklung, Probleme, Projekte, Management

Einleitung

Eine Reihe von Studien und Statistiken zeigte bereits in der Vergangenheit, dass ein Großteil aller Softwareentwicklungsprojekte scheiterte. Entweder, weil sie den Zeit- und Budgetrahmen bei weitem überstiegen, oder weil sie sogar abgebrochen wurden. Es stellt sich nun die Frage, aufgrund welcher Probleme es zu einer solch schlechten Erfolgsquote kommt? Welche Gefahren führen zu einem projekttechnischen Totalschaden und mit welchen Mitteln und Maßnahmen können diese beseitigt oder sogar vermieden werden?

In der vorliegenden Arbeit wird versucht, die bedeutendsten Hindernisse aufzuzeigen, auf die ein Entwickler bei seiner Arbeit stößt. Es wird außerdem versucht zu klären, wie es zu diesen Problemen kommt und was dagegen unternommen werden kann. Die einzelnen Schwierigkeiten werden dabei möglichst allgemeingültig und unabhängig von der eingesetzten Programmiersprache oder dem Datenbanksystem betrachtet. Da letztlich alle Probleme in einem Softwareentwicklungsprojekt sehr eng miteinander verzahnt sind, lassen sie sich nicht in voneinander getrennte Kategorien einteilen. Alle Faktoren, auch die hier nicht aufgeführten, müssen immer im Zusammenhang gesehen werden. Diese Arbeit zeigt nicht alle Stolperstellen in einem Entwicklungsprojekt auf und kann auch nicht den absoluten Leitfaden zur erfolgreichen Durchführung eines Projektes bieten. Es soll vielmehr auf Probleme hingewiesen werden, um alle Beteiligten eines Softwareentwicklungsprojektes zu sensibilisieren und einige hilfreiche Vorschläge zur Verbesserung geben. Da jedes Projekt einzigartig ist, muss jeder den für sich optimalen Weg suchen.

Grobe Anfängerfehler

Der wohl fahrlässigste Fehler, den man in der Softwareentwicklung begehen kann, ist der, sofort mit der Code-Entwicklung zu beginnen, ohne vorhergehendes Software Engineering und Systemdesign. Ursache solcher Fehler sind oft persönliche Einstellungen zum Projekt, wie beispielsweise ein falsches Einschätzen des Entwicklungsaufwandes, des Einsatzzweckes oder der Einsatzdauer.

Ein ganz ähnliches Problem besteht darin, dass die Notwendigkeit eines Vorgehensmodells nicht erkannt wird. Wer glaubt, dass „hier jeder weiß, was zu tun ist“ läuft sehr wahrscheinlich Gefahr in Planlosigkeit zu enden. Selbst in kleinen Projekten mit nur sehr wenigen Mitarbeitern können mit der Zeit Details verloren gehen, Missverständnisse auftreten, Aufgaben gedoppelt oder sogar unnütze Arbeiten erledigt werden.

Um diesen Problemen vorzubeugen, sollte man sich vorab das passende Vorgehensmodell für das Projekt suchen und als Richtschnur nutzen. Blind einem Modell zu folgen bringt allerdings auch keine Vorteile. Es ist vielmehr die jahrelange Erfahrung der Projektbeteiligten, welche die richtige Dosis und die richtige Ausgestaltung des Vorgehensmodells vorgeben. Der ganze Projektoverhead mit Anforderungsanalyse, Risikoanalyse, Qualitätsmanagement etc. erfordern zwar zusammen mehr Zeitaufwand als das eigentliche Coding, es ist aber für die erfolgreiche Durchführung eines Softwareentwicklungsprojekts unabdingbar.

Quick & Dirty Code

Zeit ist Geld, meinte schon Benjamin Franklin. Da Zeit und Kosten in der Softwareentwicklung als äquivalente Größen gelten, ist man natürlich stets bemüht, Programme möglichst schnell zu entwickeln. Genau dieser Zeitdruck führt jedoch zu niedriger Qualität. Weil das meist knappe Budget zu wenige oder ungenaue Tests vorsieht, bleiben viele Fehler lange unentdeckt. Bugs, die erst in der Produktionsumgebung auftauchen, sind um ein vielfaches teurer als wenn sie beispielsweise bereits beim Entwickler in einem Unit-Test oder in einem umfassenden Integrationstest auftauchen.

„I think there's just too much focus on "getting it out the door" that we lose focus on what we're trying to do: Create something that does its job well, and requires a minimum of maintaining.”*[3]*

Dieser Satz stammt aus einem IT-Forum und trifft das Problem auf den Kopf. Es ist fatal, ein Softwareentwicklungsprojekt nur bis zum Rollout beim Kunden zu planen und zu bewerten. Softwareentwicklungsunternehmen sind in der Regel noch viele Jahre in der Verantwortung, ob ein System wie erwartet läuft oder nicht. Genau in dieser Wartungszeit entstehen dann Aufwände, die die eine Zeitersparnis in der Entwicklung um ein vielfaches übersteigen.

Ein weiteres Problem ist eine „Nach mir die Sintflut“-Einstellung der Entwickler. Dazu kann es beispielsweise bei einer hohen Fluktuationsrate kommen. Wenn ein Entwickler weiß, dass er bald von einem Projekt abgezogen wird, kann die Motivation guten Code zu schreiben schon mal nachlassen. Der Nachfolger steht dann vor einen Trümmerfeld und wird auch nur das nötigste nachbessern.

Es gibt viele Gründe, warum Code nicht die Qualität aufweist, die er haben sollte. Deshalb ist es ein unverzichtbarer Bestandteil eines Softwareentwicklungsprojektes regelmäßig Zwischenergebnisse abzunehmen und Code Reviews durchzuführen. Eine schnellere Möglichkeit bieten automatische Code Review Tools. Diese spüren im Quellcode Muster auf, welche oft zu Problemen führen oder Sicherheitslücken darstellen. Code Review Tools können zwar Vorschläge zur Verbesserung geben, aber die endgültige Entscheidung muss trotzdem der Entwickler treffen. Ein gezieltes Qualitätsmanagement und die Benennung eines Projektqualitätssicherungsbeauftragten sind gute Methoden den genannten Problemen entgegenzuwirken. Das Qualitätsmanagement muss während des gesamten Projektes die Entwicklung begleiten. Es sollten sowohl noch nicht ausführbarer Code in Reviews, als auch ausführbarer Code in Tests, gründlich erprobt werden. Neben den Tests, in denen man überprüft, ob das System die erwarteten Werte korrekt verarbeitet, sollte man auch überprüfen, wie es auf unerwartete Werte reagiert (Fuzzing). Da nicht jedes Unternehmen über die notwendigen Kapazitäten verfügt, die ein umfassender Test erfordert, gibt es auch die Möglichkeit des Test Outsourcing. Dabei werden diese an speziell geschulte externe Teams vergeben.

Natürlich hat auch der gründlichste Test seine Grenzen. Nicht zuletzt aus Kostengründen muss die Relation zwischen Anforderungen und Aufwand gewahrt werden. Jeder Anwendungsbereich hat andere Qualitätsanforderungen. So benötigt beispielsweise ein Touristeninformationssystem bei weitem nicht die Zuverlässigkeit wie ein Lebenserhaltungssystem. Aber in der Regel gilt, die Zeit die

man heute investiere, um besseren Code zu produzieren, erspart in der Zukunft ein Vielfaches an Wartungsaufwand!

Architekturprobleme

Die Architektur eines Softwaresystems legt sowohl dessen roten Faden als auch den äußeren Rahmen fest. Werden hier Fehler gemacht, können die einzelnen Komponenten einer Software noch so raffiniert und sauber programmiert sein, das Programm wird trotzdem nur eine kurze Lebensdauer haben oder noch vor dem ersten produktiven Einsatz verworfen. Bei der Architektur müssen die verschiedensten Gegebenheiten für den Einsatz berücksichtigt werden. Dazu zählen die Nutzungsdauer, die Wiederverwendbarkeit oder die Erweiterung des Einsatzes der Software. Eine Applikation sollte so flexibel gestaltet sein, dass man sie mit möglichst geringem Aufwand an sich verändernden Anforderungen anpassen kann. Ist das nicht möglich, müssen Systeme unter Umständen komplett ersetzt werden. Es erfordert viel Erfahrung und fachliches Verständnis für die Geschäftsabläufe des Kunden, um die erforderlichen Einflussfaktoren richtig einschätzen zu können. Man sollte also möglichst einen modularen Aufbau des Systems anstreben, welcher Erweiterungen und Änderungen einfach ermöglicht. Es sollten mehrere Entwurfsvarianten vorgeschlagen und diskutiert werden.

Probleme des Entwicklerteams

Die Entwicklung von Software lässt sich nicht mit der Konstruktion von anderen Dingen vergleichen. In einem Softwareentwicklungsprojekt werden keine Materialien verbaut, die schon seit Jahrhunderten bekannt und erprobt sind. Methoden und Tools ändern sich zu schnell, um verlässliche Best Practices zu entwickeln. Bevor man genug Erfahrung hat, um eine Technik zuverlässig zu beurteilen, ist sie schon veraltet und durch eine neue ersetzt, welche aber wiederum neue, noch unbekannte Probleme aufweist. Aber gerade im Bereich der Softwareentwicklung heißt es entweder weiterschwimmen oder untergehen - einfach stehen bleiben geht nicht. Das gilt sowohl für die Entwickler als auch für die Entscheidungsträger. Wer sich nicht rechtzeitig mit neuen Technologien auseinander setzt, hat das Nachsehen. Um immer auf dem aktuellsten Stand zu sein, was ganz nebenbei gesagt auch ein entscheidender Wettbewerbsvorteil ist, sollten die Entwickler selbst, aber auch ihre Vorgesetzten, darauf achten, dass sie gezielt weitergebildet werden. Ausschließlich mit bewährten Methoden das Alltagsgeschäft zu bestreiten, bringt zwar kurzfristig die höchste Rendite, birgt aber auch die Gefahr, dass man bei neuen Herausforderungen nicht vorbereitet ist. Neben den neusten Technologien und Methoden ist aber die langjährige Erfahrung aus früheren Projekten ebenso hilfreich und wichtig.

Ein ähnliches Problem ist, wenn das Management ein auslastungsorientiertes einem qualitäts-orientierten Denken vorzieht. Nicht jeder Mitarbeiter eines Unternehmens, welches Software entwickelt, ist eine Allzweckwaffe, die man auf jedes beliebige Projekt loslassen kann. Wenn ein Java-Entwickler gebraucht wird, aber nur ein C++ Profi mit Java Grundkenntnissen zur Verfügung steht, darf man nicht erwarten, dass er die gleiche Qualität liefern kann wie in einem C-Projekt. Ganz im Gegenteil! Dieser Entwickler wäre geneigt, seine gewohnten Vorgehensweisen anzuwenden. Aber Praktiken, die in einer Programmiersprache zum optimalen Programm führen, können sich in einer anderen sehr negativ auswirken. Mitarbeiter sind nicht austauschbar wie Maschinen. Eine Verdopplung der Entwickler bewirkt keine Halbierung der Entwicklungszeit. Ganz im Gegenteil! Je mehr Mitarbeiter beteiligt sind, desto höher ist der Abstimmungsaufwand.

Die Clean Code Developer Initiative*[4]* bildet ein Wertsystem, das Prinzipien, Regeln und Praktiken für bessere Software vereint. Es kann somit für einen Entwickler ein Leitfaden sein, der ihm hilft, qualitativ hochwertigere Software zu entwickeln, die langfristig rentabler ist.

Kein gemeinsames Verständnis der Projektbeteiligten

Selbst Kollegen die schon lange zusammenarbeiten, haben nicht immer die gleiche Sicht der Dinge. Beginnend bei der Interpretation von Fachausdrücken, über Code-Style-Guides bis hin zur Namensgebung von Dateien. Wenn die im Projekt verwendetet Begriffe nicht für alle gleichermaßen verständlich in einem Projektglossar festgeschrieben sind, kann das zu Missverständnissen führen, welche den reibungslosen Ablauf eines Projektes stören. In einem Softwareentwicklungsprojekt, in dem jeder Entwickler seinen eigenen Style-Guides und Naming Conventions folgt, wird es für einen anderen Entwickler schwer, den Code seines Kollegen zu verstehen. Dadurch wird die Wartung erheblich beeinträchtigt. Unterschiedliche Dateibenennungen können dazu führen, dass die falschen Dokumente verwendet werden oder erst gar nicht gefunden werden.

Zu den wichtigsten Prinzipien für die erfolgreiche Durchführung eines Projektes zählt die Informationstransparenz. Jeder Kollege muss zu jeder Zeit wissen, was das gemeinsame Ziel ist und welche Änderungen sich vielleicht im Laufe der Zeit ergeben haben. Wenn das nicht jedem klar ist, arbeiten alle nebeneinander statt miteinander. Es müssen zu Beginn des Projektes Regeln festgelegt werden. Diese definieren wo Informationen hinterlegt werden und wie Dateien zu benennen sind, damit sie jeder schnell findet. Des Weiteren müssen die Verantwortlichkeiten festgelegt werden, wer welche Informationen wann und wohin liefern muss.

Ein gutes Mittel, um alle Projektbeteiligten auf einen Stand zu bringen, sind regelmäßige Retrospektiven. Dabei werden die Ereignisse der letzten Etappe rekapituliert und Probleme analysiert. Es wird zusammengetragen, was besonders gut lief und was die Stärken des Teams sind. Es wird aber auch überlegt, welche Probleme aufgetreten sind und wie man diese beseitigen und in Zukunft vermeiden kann. Eine konstruktive Kritik ist dabei erwünscht, aber es dürfen den Kollegen keine Vorwürfe gemacht werden. Ein solches Treffen soll schließlich das Team zusammenführen und nicht auseinander treiben. Jeder muss ermuntert werden, seine Fehler offen zuzugeben und nicht zu verschweigen. Nur wenn das Team einen respektvollen Umgang miteinander und eine angenehme Kommunikationskultur pflegt, sind die Mitarbeiter motiviert sich auch außerhalb des festgelegten Dienstweges auszutauschen. Diese Treffen bei einem Kaffee in der Kantine oder einem Bier nach Feierabend sind wichtig, damit die Mitarbeiter gut gelaunt und interessiert als Team gemeinsam das Projekt vorantreiben. Der Projektleiter hat dabei die Aufgabe den Teamspirit zu entwickeln und auf alle Beteiligten zu übertragen.

Ein weiteres Mittel zur (präventiven) Beseitigung von Unklarheiten ist das Schreiben von Berichten. Dabei geht es nicht nur darum, die eigenen Gedanken und Eindrücke für die Kollegen festzuhalten, sondern auch sich selbst zu zwingen, über das Erlebte genau zu reflektieren und im Detail zu überdenken. Mündliche Absprachen sollten unbedingt auch schriftlich festgehalten werden. Dies dient einerseits dazu, abwesende Kollegen auf dem Laufenden zu halten und andererseits, selbst keine Details zu vergessen oder um Missverständnisse, die im Gespräch vielleicht aufgetreten sind, aufzudecken. Mit all den Regulierungs- und Dokumentierungsbestrebungen darf man es aber natürlich auch nicht übertreiben, da es sonst zu ständiger Zeitnot führt und die Qualität der Ergebnisse dadurch sinkt.

Schlechte Dokumentation

Die Dokumentation ist einer der wichtigsten Bestandteile eines Softwareentwicklungsprojektes. Leider ist sie in der Praxis häufig veraltet, nicht adäquat oder fehlend. Das Schreiben einer Dokumentation ist eine sehr zeitaufwendige Arbeit. Deswegen wird der Inhalt meist auf das Nötigste reduziert, wodurch sie nur eine geringe Aussagekraft hat. Wenn die Software geändert, aber die Dokumentation nicht

angepasst wird, ist sie nicht nur wertlos, sondern kann sogar in die Irre führen. Es ist deshalb unerlässlich, die Funktionalität und Umsetzung schriftlich festzuhalten, damit das Programm später mit möglichst geringem Aufwand angepasst oder geändert werden kann.

Eine einfache Möglichkeit eine Dokumentation zu erstellen ist die, dass man die Spezifikation als Grundstein nimmt. Eine detaillierte Spezifikation enthält bereits viele Elemente, die man auch für die Dokumentation verwenden kann. Es muss aber natürlich darauf geachtet werden, dass Änderungen die sich während der Entwicklung ergeben haben, auch in die Dokumentation entsprechend einfließen. Je exakter man im Vorfeld also spezifiziert, desto einfach lässt sich das Programm nicht nur umsetzen, sondern auch dokumentieren.

Eine Möglichkeit bei der technischen Dokumentation in agilen Projekten Zeit zu sparen ist die, den Quellcode möglichst selbsterklärend zu gestalten. In einem Programm, das gut lesebaren Code und Kommentare enthält, benötigt man keine oder zumindest nur eine fachliche Dokumentation. Wenn ein Blick in den Code die Funktion und den Ablauf eines Programmes verständlich aufzeigt, ist man bei Änderungen oder Anpassungen schneller. Dokumente die den Code beschreiben, können nicht mehr veralten, da sich der Code stets selbst beschreibt.

Unklare Verantwortlichkeiten

Den Entwickler behindern leider immer wieder Verzögerungen, die entstehen, weil notwendige Entscheidungen aufgrund von unklaren Zuständigkeiten nicht getroffen werden. Wenn nicht rechtzeitig festgelegt wird, wer Projektleiter ist, wer dessen Vertreter ist und wer wem gegenüber eine Bringschuld hat, entstehen sehr schnell vermeidbare Hindernisse. Ohne zentrale Anlaufpunkte für Probleme muss sich der Entwickler mit Dingen beschäftigen, die nichts mit seiner eigentlichen Arbeit der Entwicklung zu tun haben. Solche Verzögerungen schlagen sich dann sofort in Zeit und Kosten nieder. Es muss also klar definierte Ansprechpartner geben, die die Ressourcenplanung übernehmen und den Überblick darüber behalten wer wann da ist, wem welche Informationen zugänglich gemacht werden müssen oder wo Engpässe entstehen können. Des Weiteren müssen sie mögliche Schwierigkeiten rechtzeitig erkennen und diesen entgegenwirken. Neben den projektinternen Fragestellungen ergeben sich auch immer wieder fachliche Unklarheiten. Eine Person mit dem fachlichen Know-How des Kunden muss dem Entwickler zur Verfügung stehen, um Probleme schnell und unbürokratisch zu lösen. Besonders in großen Projekten ist es schwer alle Beteiligten auf dieselben Ziele auszurichten. Dabei kann es zu Situationen kommen, in denen entweder mehrere Personen gleichzeitig oder plötzlich niemand eine Entscheidung treffen will. Sobald man beginnt den Schwarzen Peter reihum zu schieben, geht es nur noch um Politik und alle technischen Aspekte sind hinfällig.

Neben der Verantwortung, die einzelne Projektbeteiligte haben, gibt es auch noch die Verantwortung, die sich alle teilen. Damit ist gemeint, dass Probleme im Projekt nicht ignoriert werden dürfen, sondern angesprochen werden müssen. Der Mensch neigt dazu, sich mit der Zeit an Probleme zu gewöhnen. Aber das kann in einem Projekt fatale Folgen haben. Wenn tiefsitzende Schwierigkeiten nicht sofort gelöst werden und man stattdessen mit den Alltagsarbeiten fortfährt, läuft man letztendlich Gefahr, dass alle weiteren Investitionen ins Projekt völlig unnütz sind. Manchmal sieht die Lösung eines Problems eben so aus, dass man das ganze Projekt abbrechen muss. Solange dies möglichst früh geschieht, ist der Schaden noch gering. Wenn man aber zu lange wartet, wird es immer teurer. Es sollten also alle den Mut und vor allem die nötige Verantwortung haben, Probleme frühzeitig zu erkennen, anzusprechen und wenn nötig zu eskalieren. Zu den Aufgaben eines verantwortungs-bewussten Vorgesetzten gehört es dann aber auch, die Hinweise der Entwickler, also den technischen

Spezialisten, ernst zu nehmen und die richtigen Maßnahmen einzuleiten. Sobald politische Ziele über technischen und fachlichen Zielen stehen, ist der Erfolg des Projektes sehr zweifelhaft.

Kommunikationsprobleme mit dem Kunden

Was nützt der beste Techniker, wenn er nicht weiß was er tun soll? Entwickler stehen häufig vor dem Problem, dass der Kunde die Anforderungen nicht so formulieren kann, dass es der Entwickler versteht oder der Kunde weiß vielleicht sogar selbst nicht genau, was er will. Deswegen ist es wichtig, sich schon im Vorfeld mit dem Kunden zusammen zu setzen und genau zu analysieren, was er braucht. Natürlich können sich während eines Softwareentwicklungsprojektes Änderungen ergeben. Der Entwickler muss dann fähig sein, das Programm den neuen Anforderungen anzupassen. Denn die Software sollte die Anforderungen erfüllen können, die der Kunde jetzt braucht und nicht die, die er gestern wollte.

Neben der ungenügenden Spezifikation behindert den Entwickler oft auch eine ungenügende Zuarbeit des Kunden. Wenn Testdaten oder die vereinbarten Schnittstellen zu andern Systemen des Kunden nicht rechtzeitig bereitstehen, verzögert das die Entwicklung erheblich. Das schadet nicht nur dem Entwicklungsunternehemen sondern auch dem Kunden. Der Chaos-Report*[5]* zeigte bereits auf, dass die häufigsten Gründe für einen Projektabbruch unvollständige Anforderungen und mangelnde Einbeziehung der Anwender sind. *[1]* Zwischen den Kunden und den Entwicklern muss ein gemeinsames Verständnis geschaffen werden, was das Ziel ist und wie man es erreichen will. Es ist also wichtig, direkt und ehrlich mit dem Kunden zu kommunizieren. Es dürfen keine mündlichen Absprachen gemacht werden, von denen dann keiner beziehungsweise nicht alle Bescheid wissen. Sämtliche Anforderungen, Änderungen und Anfragen sollten ausschließlich über den Projekteiter gehen. Er behält den Überblick und kann die Aufgaben entsprechend verteilen.

Die enge Zusammenarbeit mit dem Kunden endet aber nicht mit der Auslieferung der Software. Genauso wichtig wie die Umsetzung des Programms ist auch die Schulung der Anwender. Wenn die Benutzer des Programms nicht mit dessen Bedienung zu Recht kommen, sind sie frustriert und mit dem Gesamtergebnis unzufrieden. Aus Sicht des Kunden ist das Projekt dann fehlgeschlagen. Eine gute Referenz und eine Beauftragung von weiteren Entwicklungen darf man sich von so einem Kunden also nicht erhoffen.

Schlechtes Management

Die ersten Fehler in einem Projekt werden oft schon begangen, bevor überhaupt ein Projektteam zusammengestellt wird. Es werden beispielsweise Fixpreisprojekte angeboten, obwohl noch nicht einmal ein entsprechendes Pflichtenheft vorliegt. Es werden Preise und Angebote gemacht, ohne genau zu wissen, was der Kunde eigentlich will oder noch viel wichtiger, was er wirklich braucht. Der Entwickler steht dann vor einer mangelhaften Spezifikation und weiß nicht genau, was entwickelt werden soll. Wenn bei der Aufwandsschätzung nicht die entsprechenden technischen Spezialisten hinzugezogen werden, kommt es zu unhaltbaren Zeitvorgaben. Ähnliche Probleme entstehen, wenn während der Entwicklung noch weitere Anforderungen angenommen werden, obwohl die Auslastung bereits am Limit ist. All diese Probleme führen letztlich dazu, dass der Entwickler stets unter übermäßigem Zeitdruck steht, was natürlich auch die Qualität seiner Arbeit negativ beeinflusst. Aber Qualität darf nicht verhandelbar sein! Wenn Kundenwünsche angenommen werden, wohl wissend, dass sie nicht brauchbar oder nicht umsetzbar sind, nur weil man nach Aufwand bezahlt wird, ist das für ein Softwareentwicklungsunternehmen langfristig nicht rentabel. Schlecht funktionierende und unwartbare Software bedeuten den Totalschaden für ein Projekt.

Neben einer Anforderungsanalyse gehört auch eine Risikoanalyse zu den wichtigsten Dingen die man vor dem Start eines Projektes durchführen sollte. Aufgrund eines hohen Kosten- und Termindrucks und einer fehlenden Datenbasis mit brauchbaren Erfahrungswerten können oft nicht alle Risiken erfasst werden. Wenn aber Risiken nicht beachtet oder falsch bewertet werden, kann das katastrophale Folgen für das Projekt haben. Ein gutes Risikomanagement ist also unverzichtbar. Es geht hierbei aber nicht um eine exakte Wissenschaft, sondern um die Fähigkeit bisher unbekannte Faktoren zu identifizieren und positive Chancen zu erhöhen und negative Risiken zu minimieren.

Das Bindeglied, das alle Projektbeteiligten zusammenführt und die Maschenerie der Entwicklung am Laufen hält, ist das Projektmanagement. Es hat die zentrale Aufgabe der initialen Planung und der täglichen Koordination. Der Projektmanager ist dafür verantwortlich, dass allen Projektbeteiligten die erforderlichen Informationen zugänglich gemacht werden. Absprachen zwischen dem Kunden und dem Management des Entwicklungsunternehmens müssen dem Entwicklerteam entsprechend kommuniziert werden, da es sonst zu Gerüchten und Verunsicherungen kommen kann. Wenn das Entwicklerteam den nötigen Überblick verliert, könnten sich wichtige Details in der Architektur oder der Implementierung für den weiteren Projektverlauf unvorteilhaft entwickeln. Nur wenn man frühzeitig und vorrausschauend planen kann, lassen sich aufwendige Änderungen und Anpassungen vermeiden. Wenn keine gemeinsamen Regeln eingehalten werden und jeder Beteiligte nur die Vorgaben einhält, die ihm vorteilhaft erscheinen, endet das Projekt schnell im Chaos.

Fazit

Jedes Projekt hat seine Eigenarten. Dafür sorgen sowohl branchenspezifische als auch anwendungsspezifische Einflüsse. Die einzelnen Faktoren wie Staffing, Vorgehensmodell, Kunde und Ziel müssen zusammenpassen. Dabei ist die richtige Wahl und Dosierung der eingesetzten Mittel und Methoden immer ein Drahtseilakt. Man muss schon vor dem Entwicklungsbeginn den Weg richtig planen, sonst gerät das Projekt schnell von der Straße ab. Wenn man dabei zu starr plant, trifft einen der Zufall härter. Immer einen Plan B und C vorzuhalten, ist jedoch viel zu aufwendig. Alle Projektbeteiligten müssen also aufmerksam genug sein, um Probleme rechtzeitig zu erkennen, zu melden und gemeinsam mit den anderen Projektbeteiligten zu beseitigen. Die Entwickler brauchen dafür aber auch entsprechende Befähigungen, um Probleme gegebenenfalls eskalieren zu können. Sie dürfen bei ihren Vorgesetzten nicht auf taube Ohren stoßen. Manager müssen akzeptieren, dass sie nur für die Organisation um das Projekt verantwortlich sind. Sie dürfen keine technischen und fachlichen Entscheidungen treffen, nur um politische Ziele, wie die Gewinnung eines strategischen Kunden, zu verfolgen. Für das Projekt würde das sehr wahrscheinlich negative Folgen haben. Eine technische Beurteilung und Aufwandsschätzung sollte immer durch den entsprechenden Fachexperten erfolgen, wobei Qualität nicht verhandelbar sein darf. Die Zeit, die man bei der Entwicklung zu sparen versucht, wird man als Vielfaches später in die Nacharbeit stecken müssen. Ein Entwickler wünscht sich letztendlich nur klare Strukturen, Verantwortlichkeiten und einfache Richtlinien für das Projektmanagement, die sich konsequent ohne zu viel Aufwand durchsetzen lassen. Alle Projektbeteiligten sitzen gemeinsam an einem Projekt und sind als Team dafür verantwortlich, es zu einem erfolgreichen Ergebnis zu lenken.

Quellen und Literarturempfehlungen:

[1] A. Schatten: Best Practice Software-Engineering – Spektrum Akademischer Verlag, Heidelberg 2010
[2] T. Grechenig: Softwaretechnik – Pearson Studium, München 2010
[3] http://stackoverflow.com/questions/75771/what-is-the-biggest-problem-with-software-development

[4] http://www.clean-code-developer.de/
[5] http://net.educause.edu/ir/library/pdf/ncp08083b.pdf

Kontaktadresse:

Nico Zinner
Trivadis GmbH
Industriestraße 4
D-70565 Stuttgart

Telefon: +49-711-903 63 230
Fax: +49-711-903 63 259
E-Mail nico.zinner@trivadis.com
Internet: www.trivadis.com