

MAGAZIN FÜR SOFTWARE-ENTWICKLUNG

swdevelopment

SCHWERPUNKT:

Globale Suche über Web-Anwendungen

SUCCESS-STORY:

Busarchitektur für Applikations-Schnittstellen

SCHWERPUNKTTHEMEN:

TEST THE WEB

DIE ZÄHMUNG DES WEB

FACHTHEMEN:

Entwicklung von Softwarekomponenten mit dem MTS

SOFTWARE-ENTWICKLUNGSPROZESS:

Der Untergang der Titanic oder der Weg aus der Krise?

Entwicklung ist Integration

ISSN 1616-4970

SONDERDRUCK FÜR

ARS NOVA Software GmbH

FRANK LEDERMANN

SOFTWARE-ENTWICKLUNGSPROZESS: DER UNTERGANG DER TITANIC ODER DER WEG AUS DER KRISE?

Dank modernster und ausgefeilter Technologien gleicht die Realisierung von komplexen Softwareprojekten der Atlantiküberquerung mit einem gigantischen, unsinkbaren, schnellen und mit allem Komfort ausgestatteten Luxusliner. Der Projektleiter steht auf der Brücke, das Ruder fest in der Hand und dirigiert mit einer Vielzahl von Knöpfen und anhand diverser Messinstrumente eine Heerschar von dienstbaren Geistern. Anforderungen werden zumindest halbautomatisch in Use Cases umgesetzt, Tools generieren aus dem erstellten Design ablauffähigen Programmcode, selbst die Oberflächen generieren sich mit Werkzeugunterstützung fast von selbst. Das Mapping von Objektstrukturen auf relationale Datenbanken geschieht vollautomatisch, ein bisschen Integration und Test, geplant mit entsprechender Softwareunterstützung, das war's.

Weit und breit kein Sturm in Sicht, gelegentlich auftretende kleinere Böen werden hochtechnologieunterstützt ausgetrimmt. Die besten Voraussetzungen für einen ruhigen Schlaf der Passagiere, die sich gleichermaßen aus Kunden und Vorgesetzten zusammensetzen.

Und dennoch erleiden gut 80% dieser Kolosse eine Kollision mit den unter Wasser lauenden Eisbergen der Realität - die Folge sind versenkte, bestenfalls mit großem Aufwand reparable Projekte.

Mit den Eigenschaften dieser "Eisberge" und den Möglichkeiten, diese zu umschiffen, befasst sich nachfolgender Artikel.

EINFÜHRUNG

Wir verfügen heute über das nötige Wissen und die nötigen Erfahrungen für eine erfolgreiche Abwicklung von Softwareprojekten unterschiedlichster Art. Doch es gibt wohl kaum ein Gebiet der Informationstechnologie, auf dem die bedenkliche Kluft, die stets zwischen dem theoretischen Wissen einerseits und seiner praktischen Anwendung andererseits besteht, größer und schwerer zu überbrücken wäre.

In den vergangenen zwei bis drei Jahrzehnten haben wir zahlreiche, äußerst bedeutende Veränderungen in der Technologie und deren Einsatz erlebt. Aber weit weniger gravierend sind die Veränderungen in der konkreten Abwicklung von Softwareprojekten. Die Rahmenbedingungen unserer Projekte haben sich, insbesondere in den vergangenen zehn Jahren, durch das heterogene Umfeld und Anforderungen an stark verkürzte Entwicklungszeiten in hohem Maße verändert. Die über den Erfolg eines Projekts entscheidenden Faktoren jedoch sind, wenn man situationsbedingt angepasste Prioritäten außer Acht lässt, immer noch dieselben: Zeit, Inhalt, Kosten.

Heute haben wir die Wahl zwischen verschiedenen, mehr oder weniger ausführlich spezifizierten, schwergewichtigen und leichtgewichtigen, konkreten und abstrakten Entwicklungsprozessen (sogenannten Prozessframeworks). Sie enthalten teilweise entgegengesetzte, aber nicht selten auch weitgehend übereinstimmende Leit motive und Elemente. Es gibt genügend Publikationen, die die sogenannten Erfolgsfaktoren von OO-Projekten dokumentieren: iterativ und inkrementell, phasenorientiert, Use Case basiert, architekturzentriert, risikoorientiert, etc. Wir alle haben diese Begriffe häufig genug gehört und gelesen und mit einer Ausnahme sollen sie an dieser Stelle nicht erneut vertieft werden. Es ist notwendig, auf einige übergeordnete Charakteristika erfolgreicher Projektrealisierung einzugehen.

Die oben genannten verschärften Rahmenbedingungen ziehen nicht selten annähernd die gesamte Aufmerksamkeit und Energie der Projektteams auf sich, um die hoch komplexen Technologien und Methoden zumindest einigermaßen sinnvoll einzusetzen und um mit ihnen gerade noch brauchbare Ergebnisse zu erzielen. Dies führt immer häufiger zur Vernachlässigung der klassischen Elemente des Managements, die paradoxerweise gerade mit den neuesten Erkenntnissen zur Methodik der Softwareentwicklung an Bedeutung gewinnen.

DIE NEUE ALTE HERAUSFORDERUNG AN DAS MANAGEMENT

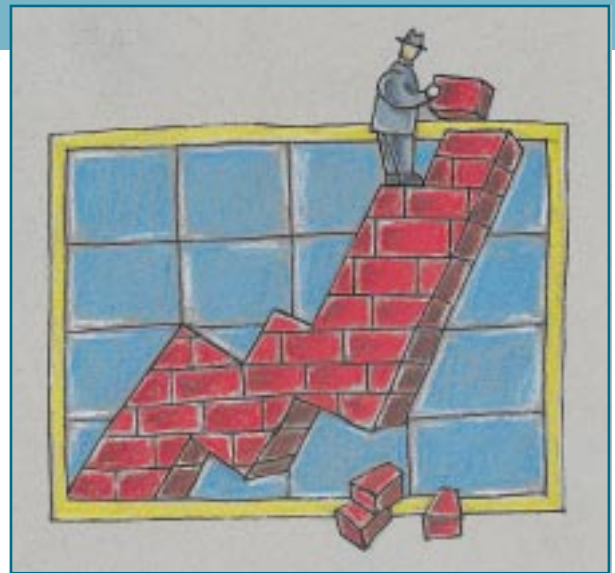
Zu den wichtigsten Aufgaben eines Projektmanagers gehört es, Kommunikationsbarrieren zu durchbrechen und Kooperationskompetenz zu fördern. Dies sind nicht nur offensichtliche Voraussetzungen für das so häufig postulierte risikobewusste Vorgehen. Welcher Mitarbeiter hat schon Freude bei der Arbeit, wenn innerhalb seines Projektteams eine Politik der Informationsbewachung und des Abschottens möglicher Einflüsse auf die Tätigkeit eines Einzelnen herrscht? Als bestes Beispiel und Vorbild sollte der Projektmanager selbst dienen. Beziehen Sie Ihre Mitarbeiter in Entscheidungsprozesse ein. Besprechen Sie die einzelne Aufgabenstellung mit potenziellen Beteiligten und seien Sie aufgeschlossen für Verbesserungsvorschläge. Für Sie als Projektleiter muss genau das die höchste Bedeutung haben, was Sie (noch) nicht wissen.

In großen Produktionsbetrieben reichte es früher vielleicht aus, einfache Handgriffe oft genug auszuführen, um ein zufriedenstellendes Tagesergebnis zu erzielen. Mitdenken war dabei nicht nötig - und gelegentlich nicht einmal erwünscht. Heute wird Managen nicht mehr verstanden als die Kunst, Aufgaben an Mitarbeiter zu verteilen und den Vorgang der Umsetzung zu beaufsichtigen und zu kontrollieren. Projektleitung ist weit mehr als Führen und Leiten von Menschen. Es müssen Prozesse entworfen und in die Praxis umgesetzt werden. Die heutige Komplexität und Anforderung an schnelle Reaktion kann nur beherrscht werden durch Eigenverantwortung und Kompetenz aller Beteiligten, sowie durch geeignete Prozesse. Dabei sind die Werkzeuge, die mit den Prozessen eingeführt werden, an sich nicht produktiv - kein Werkzeug ist von sich aus produktiv. Nur der Mensch ist in der Lage, sich selbst und verfügbare Werkzeuge gewinnbringend einzusetzen.

Über den Start eines neuen Projekts entscheidet oftmals der Faktor Zeit. Vor allem im typischen Fall der Beteiligung von externen Softwarefirmen ist die Zeitdauer zwischen Ausschreibung, Angebotsfrist und vorgegebenem Starttermin meist sehr knapp bemessen. Wer hier lange eine Lösung suchen muss, wie das Projekt zu organisieren ist und immer aufs Neue Vorgänge entwickelt und Ergebnisse spezifiziert, hat denkbar schlechte Aussichten auf Erfolg. Ein eingespielter und in der Praxis mehrfach bewährter Entwicklungsprozess unterstützt bei einem schnellen Projektstart. Selbst, wenn kein Projekt dem anderen gleicht. Bestimmte Elemente, wie beispielsweise Planungs- und QS-Maßnahmen wiederholen sich mit Sicherheit.

Ferner hat die Dauer eines Projekts erheblichen Einfluss auf den Projektverlauf. Denn, wenn wir über die Zukunft auch nichts wissen, so wissen wir doch, dass ihre Risiken in geometrischer Progression wachsen, je langfristiger wir sie zu prophezeien und vorherzubestimmen suchen. Folgendes Beispiel aus der Mechanik und Kräftelehre veranschaulicht auf etwas andere Weise die Motivation für inkrementelles Vorgehen: Stellen Sie sich einen Tankklaster vor, dessen Tank halb gefüllt ist. Erfährt der Laster einen Impuls in eine bestimmte Richtung, so wird die Trägheit der flüssigen Ladung dazu führen, dass sie sich entgegen der Impulsrichtung innerhalb des Tanks verlagert. Ein Tankklaster wäre nicht straßentauglich, wenn der Tank nicht speziell für diesen Fall konstruiert wäre. Sein Inneres ist nämlich in mehrere Parzellen gegliedert, die eine unerwünschte Verlagerung des gesamten Inhaltes verhindern und damit die Stabilität des Tankklusters sichern. Jede Parzelle nimmt einen kleinen Teil der Belastung auf und minimiert somit den Störeinfluss.

Mehrere Studien belegen, dass sich innerhalb von nur einem Jahr durchschnittlich ca. 15%-20% aller Anforderungen an ein Softwaresystem ändern (vgl. [1]). Wer nicht in der Lage ist, mit dieser Dynamik umzugehen, wird sein Projekt schwerlich zum erfolgreichen Abschluss führen können. Gerade dieser Aspekt wird durch die schrittweise Vorgehens-



weise (inkrementell) mit wiederholten Arbeitsschritten (iterativ) besonders gut berücksichtigt. Es ist beinahe die entscheidende Neuerung moderner Entwicklungsprozesse, immanent spontan eintretende Fakten bewusst zu behandeln und mit vorgesehenen Maßnahmen zu reagieren. Als Beispiele hierfür seien an dieser Stelle Änderungsmanagement, regelmäßige Integration, regelmäßiges Testen, regelmäßige Demonstration für Anwender und potenzielle Planungsanpassung an Iterationsgrenzen genannt.

ARBEITSTEILUNG UND SINN FÜRS GANZE

Wenn Sie als Projektleiter die Wahl hätten, Ihr Projektteam aus einer Gruppe von "Allroundern" oder ausgewählten Spezialisten zusammenzusetzen, welche Wahl würden Sie treffen? Das bekannte Beispiel des Orchesters und des Dirigenten bringt hierfür einen entscheidenden Aspekt zum Ausdruck: Jeder Dirigent wird bemüht sein, die besten Musiker für genau diejenigen Instrumente auszuwählen, die für das Musikstück benötigt werden. Es stellt für den Dirigenten überhaupt kein Problem dar, dass etwa ein Klarinettist nie gut Geige spielen wird oder dass ein Hornist sich nie gerne einer Oboe zuwenden wird. Selbst ein Wechsel innerhalb einer Instrumentengruppe, beispielsweise innerhalb der Blasinstrumente oder Streichinstrumente, kommt nicht in Frage. Der Dirigent braucht genau diese Spezialisten und er wird daher nie von einem Geiger verlangen, dass dieser auch noch Trompete spielt. Er wird aber eines tun: sich alle Mühe geben, den Musikern das Musikstück **als Ganzes** verständlich zu machen und er verlangt von jedem Instrumentalisten, dass er sich im Hinblick auf das Ganze ins Orchester integriert.

Nun soll dies kein Aufruf zum hochspezialisierten Expertentum sein. Die Vielseitigkeit eines "Allrounders" ist genauso gefragt, wie die "Exzellenz" eines Spezialisten. Es darf nicht vergessen werden, dass es bei der Produktion von Software darauf ankommt, die nötigen Arbeitsschritte effizient durchzuführen und Ergebnisse von hoher Qualität zu erzielen. Für besonders schwierige und knifflige Probleme in Teilbereichen liegt es nahe, den Spezialisten zu Rate zu ziehen. Beispielsweise, wenn ein Konzept für eine hoch verfügbare verteilte E-Business-Applikation

mit sehr großem Transaktionsvolumen erstellt werden soll. Wenn es dagegen darum geht, einen bestimmten Entwicklungsschritt durchzuführen, der verschiedene Einzelfertigkeiten verlangt, ist genau derjenige gefragt, der alle benötigten Kenntnisse in sich vereint. Zum Beispiel kann es schon bei der Erweiterung einer einzelnen Anwendungsmaske um ein zusätzliches Eingabefeld erforderlich sein, in sämtliche Schichten der Softwarearchitektur einzugreifen, wie etwa Oberflächenschicht, Domänenschicht, Anwendungsschicht und Datenbankschicht.

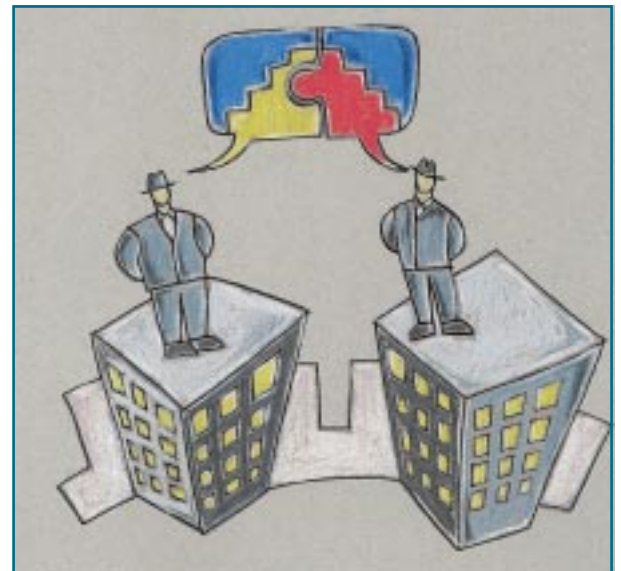
Es ist wohl angebracht, zu hinterfragen, ob wir diese Aspekte in unseren Softwareprojekten ausreichend berücksichtigen. Dabei ist es doch, wenn wir nur eine Minute darüber nachdenken, fast selbstverständlich, dass wir die besten Ergebnisse nur dann erzielen können, wenn wir von unseren Aufgaben, den verfügbaren Mitteln und einer gewählten Lösungsstrategie überzeugt sind. So ist es beispielsweise naheliegend, dass ein Datenbankexperte Datenbank-Modelle entwickelt und optimiert, während ein Experte für ergonomische Anwendungsoberflächen Interaktionskonzepte und grafische Benutzungsoberflächen entwirft. Wer als Projektleiter bewusst umgekehrt handelt, muss erstens verifizieren, ob die betroffenen Mitarbeiter Interesse an der neuen Herausforderung haben - zumindest dieser nicht abgeneigt sind - und zweitens die Ausbildungskosten in Zeit- und Aufwandsbudget einschließlich Risikopuffer berücksichtigen.

RESULTATORIENTIERUNG

Eine in der Managementliteratur gerne erzählte Geschichte ist die von den drei Maurern. Auf die Frage, was sie täten, antwortete der erste: "Ich verdiene hier meinen Lebensunterhalt." Der zweite sagte, während er weiterarbeitete: "Ich bin der beste Maurer des Landes." Der dritte blickte strahlenden Auges von seiner Arbeit auf und erklärte: "Ich helfe mit, eine Kathedrale zu bauen." Welcher von den dreien am besten auf das gemeinsame Resultat ausgerichtet handelt, ist eine rhetorische Frage.

Um auf das zuvor genannte Beispiel des Orchesters zurückzugreifen, ist zu sagen, dass die Aufgabe der Gruppe der Instrumentalisten sich in einem wesentlichen Punkt von der eines Projektteams unterscheidet. Alle Mitglieder des Orchesters sind dafür verantwortlich, im Sinne des Musikstücks harmonisch zusammen zu spielen. Bei der Softwareentwicklung geht es darum, Ergebnisse zu erarbeiten. Um den Erfolg eines Softwareprojekts zu gewährleisten, ist es nötig, dass jede einzelne Tätigkeit auf die Zielsetzung des Ganzen ausgerichtet ist. Als Endprodukt entsteht immer ein bestimmtes neues oder verändertes Programm - eine bestimmte Anwendung. Häufig handelt es sich dabei um Individualsoftware. Dabei ist es die Aufgabe des Projektleiters, allen Beteiligten den Sinn für das Ganze zu vermitteln und ihren Beitrag zur Erreichung des Projektziels in allen Bereichen deutlich zu machen. Es leuchtet ein, dass nicht jeder Einzelne einen unmittelbaren Beitrag für sämtliche Bereiche eines Projektziels leistet. So wird beispielsweise der Beitrag eines Datenbankschemas auf den Aspekt Ergonomie der Benutzungsoberfläche keinen unmittelbaren Einfluss nehmen. Sehr wohl jedoch auf Performanz und Wartbarkeit. Doch wenn von einer Gruppe oder von einem einzelnen Mitarbeiter nicht verlangt wird, dass er einen Beitrag zu einem der Bereiche leistet, dann sollte dies auch klar zum Ausdruck gebracht werden.

Dass es nicht einfach ist, kohärente Einzelergebnisse zu erzielen, mögen zahlreiche Projektleiter vielfach schmerzlich erlebt haben. Angenommen, es befassen sich zwei Projektmitarbeiter mit der Erstellung jeweils eines Designs für eine Stücklistenverwaltung und für eine Auftragsverwaltung. Beide Mitarbeiter haben auf Grund unterschiedlicher Erfahrung und auf Grund unterschiedlichen Hintergrundwissens eine eigene Vorstellung, wie genau ein Design auszusehen habe. Aller Wahrscheinlichkeit nach werden sehr unterschiedlich gestaltete und damit nicht vergleichbare



Ergebnisse resultieren, selbst wenn beide Ergebnisse für sich betrachtet vollständig und korrekt sein mögen. In einem komplexen Realisierungsprojekt ist also großer Wert darauf zu legen, Ziele eindeutig genug zu spezifizieren und zu vereinbaren, um entsprechende Resultate zu erhalten.

Als Projektleiter sind Sie in einer guten Situation, wenn Sie über einen etablierten Software-Entwicklungsprozess verfügen. Denn er unterstützt die vereinte Anstrengung aller Projektmitarbeiter, zielgerichtet zu handeln, da er Zwischenziele eindeutig definiert, Abhängigkeiten explizit darstellt und einheitliche Verfahren für wiederkehrende Aufgaben spezifiziert. Wird ein sinnvoller Software-Entwicklungsprozess richtig genutzt, ist sichergestellt, dass immer klare Ziele bekannt sind. Auch, wenn sich diese im Verlauf der Projektzeit ändern. Jeder Mitarbeiter sollte auf die Frage: "Wie beschreiben Sie Ihre aktuelle Tätigkeit/Aufgabe im Projekt?" auf ähnliche Weise wie die folgende antworten können:

- "Ich erstelle für unseren Kunden das (Teil-)Ergebnis ‚xyz‘ (auf Grundlage des Ergebnisses ‚abc‘ als Voraussetzung für ‚def‘)."
- "Ich verifiziere das (Teil-)Ergebnis ‚xyz‘ bezüglich aller funktionaler und/oder aller nicht-funktionaler Anforderungen unseres Kunden."

Anstatt: "Ich füge eine weitere Schaltfläche in diese Maske ein (weiß aber nicht, wozu diese benötigt wird und kümmere mich auch nicht darum)." Soll es also besser heißen: "Ich stelle dem Anwender die gewünschte Funktion zusätzlich per Schaltfläche zur Verfügung, weil der Aufruf der häufig benötigten Funktion über das Untermenü zu umständlich ist."

Die beiden zentralen Aspekte, die hier ins Blickfeld gerückt werden sollen, sind: Es kommt auf **Ergebnisse** an, die **für den Kunden bzw. Anwender** produziert werden. Diese Ergebnisse werden unter Berücksichtigung bestehender Ergebnisse oder Anforderungen erstellt, um wiederum als Input für weitere Aktivitäten zu dienen.

Sämtliche Aufgaben und Tätigkeiten innerhalb eines Software-Entwicklungsprojekts müssen diesen Charakter tragen, um wirksam zum Erfolg des Projekts beizutragen.

Für diejenigen unter Ihnen, die obige Antworten als selbstverständlich ansehen, sei angemerkt, dass sich die Erstellung von Individualsoftware im Gegensatz zur industriellen Produktion von Massengütern, beispielsweise Bremscheiben für Automobile, praktisch nie in gleicher Art und Weise wiederholt und dass hierdurch ein hoher Spannungsbogen zwischen geforderter Kreativität und professioneller Routine wirkt. Wer dennoch meint, dies - die Resultatorientierung - sei trivial, dem empfehle ich, diese Frage seinen Mitarbeitern einfach einmal zu stellen.

SELBSTKONTROLLE

Eigenverantwortliches Handeln der Mitarbeiter, das, wie bereits erwähnt, immer dringender benötigt wird, setzt Wille und Kompetenz für eigenverantwortliche Entscheidungen voraus. Dies schließt das Tragen der Verantwortung für die Konsequenzen eigenverantwortlicher Entscheidungen mit ein. Die Basis hierfür ist Selbstkontrolle. Bereits 1955 hat Peter F. Drucker, der Erfinder des "Management by objectives" in seinem Buch "The Practice of Management" diese Bedeutung klar gesehen (vgl. [2]). Er spricht nicht nur von "Management by objectives" sondern von "Management by objectives **and self-control**". Das folgende Beispiel verdeutlicht die Rolle und Bedeutung von (Selbst-) Kontrolle: Ein Ruderer ist alleine ohne äußeren Bezugspunkt und hat die Aufgabe, einen Hafen zu erreichen. Er kennt die Koordinaten des Hafens und hat die Möglichkeit, seine eigene Position mittels Kompass und die Richtung, in der der Hafen liegt, festzustellen. Er kann entweder seine Position und die Richtung, in der der Hafen liegt, feststellen oder rudern, um voranzukommen. Es leuchtet ein, dass er in sinnvollen Abständen seine Position kontrollieren und seine Richtung anpassen muss, um weder Kraft noch Zeit zu verschwenden und das Ziel zu erreichen. Kontrolliert er nicht, verschwendet er mit großer Sicherheit Kraft oder er kommt womöglich nie an. Kontrolliert er sehr häufig, verschwendet er mit Sicherheit Zeit.

Kontrolle - ob Selbstkontrolle oder Kontrolle von außen - muss sein und sie muss in Bezug auf das Kosten-Nutzen-Verhältnis bewertet werden. Die meisten Menschen kontrollieren nicht gerne und lassen sich noch weniger gerne selbst kontrollieren. Sie fühlen sich in ihrer Freiheit und Kreativität eingeschränkt. Doch, ob und wo Freiräume und Kreativität benötigt werden, hat nichts mit Kontrolle zu tun. Sie haben viel mehr mit Organisation zu tun. Selbst wenn ein Maximum an Freiraum und Kreativität benötigt würde, kann auf Kontrolle nicht verzichtet werden. Denn, wie soll man sonst feststellen, ob erstens die Freiräume tatsächlich existieren und zweitens diese auch tatsächlich genutzt werden. Natürlich darf Kontrolle nicht zum Selbstzweck werden und ganz entscheidend ist, dass Kontrolle immer in Verbindung mit sinnvollem Feedback stattfindet, so dass der gewünschte Steuerungs- und Lerneffekt erzielt wird. Die Frage ist also nicht, ob Kontrolle sinnvoll ist, sondern wie kontrolliert werden sollte.

Bei der Entwicklung von Softwaresystemen können parallel unterschiedliche Möglichkeiten zur Kontrolle genutzt werden. Bevor ich einige von ihnen benenne, ist zunächst zu bestimmen, was denn tatsächlich kontrolliert werden muss. Allgemein formuliert sind alle Resultate in Verbindung mit den benötigten Ressourcen (Zeit, Aufwand, Kosten) zu kontrollieren. D.h. zum Beispiel, jedes einzelne Ergebnisdokument (Use Case Modelle, Use Case Beschreibungen, Designdokumente, Entwürfe der Anwendungsoberflächen, etc.) und jede neu implementierte oder veränderte Komponente. Schon während der Erstellung von Ergebnissen ist es für jeden Mitarbeiter möglich, die Qualität seiner Ergebnisdokumente und seines Programmcodes selbst zu kontrollieren. Werkzeuge (z.B. automatische

CodeChecker oder mit Junit programmierte Testfälle), Richtlinien (z.B. Java, XML und HTML Programmierrichtlinien, Oberflächenstyleguides und Architekturstyleguides für verteilte Anwendungen mit EJBs) und Vorlagen (z.B. für Use Case Beschreibungen und Modelle oder technische Dokumentationen) unterstützen ihn bei der Selbstkontrolle.

Möglicherweise kennen Sie den Fall eines Fachkonzepts (= fachliche Spezifikation eines Ausschnitts aus dem Anwendungsbereich), in dem Begriffe aus der Fachwelt mit ihren relevanten Eigenschaften beschrieben werden. Es liegt nahe, jeden fachlichen Begriff mit einer kurzen Beschreibung zu versehen und eine (häufig zunächst unvollständige) Liste fachlich relevanter Eigenschaften jedes Begriffs aufzuzählen. Wenige denken jedoch bereits bei der fachlichen Analyse daran, Kardinalitäten, gültige Wertebereiche und Standardwerte (sogenannte Defaults) zu benennen. Gelegentlich erkennt man erst viel später, im schlimmsten Fall erst während der Implementierung, dass noch unklar ist, welche Eigenschaften statisch sind, automatisch berechnet oder vom Anwender eingegeben werden. Zudem können unerwartete, weil während der fachlichen Analyse nicht beachtete Mengengerüste die Performanz des entworfenen Systems empfindlich beeinflussen, so dass sogar ein grundlegendes Re-Design nötig ist. Die Folge sind bestenfalls überhöhte Aufwände, meistens jedoch auch Terminverschiebungen, die mit dem Kunden und Anwender vereinbart werden müssen. Mit einer einfachen Maßnahme kann das Risiko, dass einer der oben genannten Problemfälle eintritt, minimiert werden. Wenn verschiedene Arten von Ergebnisdokumenten spezifiziert sind (Ergebnistypen), ist die Gliederung jedes Ergebnisdokuments bindend vorgegeben. Dies gewährleistet nicht nur ein einheitliches Erscheinungsbild der Dokumente, es hilft auch sicherzustellen, dass alle relevanten Gesichtspunkte bei der Bearbeitung eines bestimmten Themas betrachtet werden.

Nach der Frage, was zu kontrollieren ist, ist als nächstes zu klären, ob lückenlose Kontrollen nötig sind oder Stichproben ausreichen. Es ist leicht einzusehen, dass eine lückenlose Kontrolle, beispielsweise jeder Programmcodezeile eines mehrere tausend Objektklassen und mehrere zehntausend Methoden umfassenden Programms inef-



fizient wäre. Hier sind Stichproben an signifikanten Stellen angebracht. Dasselbe gilt für sehr umfangreiche Ergebnisdokumente, die hunderte Seiten umfassen, - was im Übrigen darauf hindeutet, dass man sich bei der Dokumentation eventuell nicht auf das Wesentliche beschränkt hat. Allerdings muss Folgendes ohne Ausnahme vollständig kontrolliert werden: unerledigte Angelegenheiten - sogenannte offene Punkte. Wer nicht lückenlos dafür sorgt, dass Dinge die vereinbart sind, nicht vergessen oder übersehen werden, hat sein Ziel schon zur Hälfte verfehlt. Es gibt kaum eine Möglichkeit, Vertrauen und Glaubwürdigkeit leichter einzubüßen, als Termine zu versäumen und zugesagte Dinge unter den Tisch fallen zu lassen (vgl. [4]). Das heißt natürlich nicht, dass immer alle Dinge auch erledigt werden. Aber eine Sache soll nicht deshalb nicht erledigt werden, weil sie vergessen oder übersehen wurde, sondern weil man es so entschieden hat. Wie dies genau gemacht wird, kann im Einzelfall unterschiedlich sein. Manche führen eine Art Tagebuch, andere verwenden ein Programm zur Verwaltung von Tabellen, und schließlich gibt es die PDAs. Unabhängig, welches Ihre bevorzugte Alternative sein mag, regelmäßige Überprüfung und Priorisierung der unerledigten Angelegenheiten sind in jedem Fall erforderlich. Und dies muss mit großer Sorgfalt geschehen, denn oftmals sind es die scheinbar kleinen Dinge, die schwerwiegende Konsequenzen nach sich ziehen.

ZEITEINTEILUNG

"Man verliert die meiste Zeit damit,
daß man Zeit gewinnen will."

- John Ernst Steinbeck

"Mit welchem Aufwand ist für die Realisierung dieser Aufgabe zu rechnen?" Mit dieser Frage kann ein Projektleiter den Zähler einer Zeitbombe starten, denn sie setzt den Gefragten zwangsläufig unter Druck. Er geht vermutlich davon aus, dass die Aufgabe so schnell wie möglich zu realisieren ist - und liegt damit wahrscheinlich richtig - und möchte dem Projektleiter entgegenkommen. Darüber hinaus ist der gefragte Mitarbeiter eventuell nicht in der günstigen Situation, dass er bereits mehrere ähnliche Aufgabenstellungen bearbeitet hat. Er überschlägt die aus seiner Sicht unbedingt nötigen und für ihn oberflächlich sichtbaren Schritte und nennt dem Projektleiter einen Aufwand, den er als Basis für seine Planung verwendet. Erst viel später stellen beide fest, dass für essenzielle Dinge keine Zeit vorgesehen ist: signifikante Testfälle wurden nicht berücksichtigt, die Zeit für die Erstellung der Dokumentation fehlt und eine aufwendige Analysephase sprengt den Zeit- und Aufwandsplan. Wichtige Aufgaben, die regelmäßig die Zeit des Mitarbeiters beanspruchen, hat der Projektleiter bei seiner Terminkalkulation nicht berücksichtigt. Die ursprüngliche Schätzung betrachtet im Wesentlichen nicht mehr als die reine Implementierung mit einem kleinen Sicherheitspuffer und beide haben den Abschlusstermin nur nach dem aktuellen Datum und dem geschätzten Aufwand berechnet.

Verschiedene Studien belegen, dass die reine Implementierung je nach Projekt nicht mehr als 30-50% des Gesamtaufwands ausmacht. Ähnlich verhält es sich mit der Zeitdauer (vgl. [3]). Der Projektleiter im Beispiel oben machte einen entscheidenden Fehler. Bei seiner Planung versäumte er es, sicherzustellen, dass der Mitarbeiter **alle** benötigten Vorgänge und Ergebnisse berücksichtigt. Er hat mit der Erstellung des Projektplans die Aufgabe, das Projekt unter Berücksichtigung der Rahmenbedingungen zu definieren. Der Projektplan dient als Richtschnur

für den Projektverlauf und als Maßstab für den Projekterfolg. Um den Projektverlauf zu definieren, sind alle Vorgänge zu bestimmen, in sinnvoller Beziehung zueinander anzuordnen und den benötigten Ressourcen zuzuordnen. Hierbei müssen auch regelmäßig wiederkehrende Aufgaben, wie beispielsweise Statusberichte, Teamsitzungen, etc. berücksichtigt werden. Es reicht zunächst aus, die Dauer der einzelnen Vorgänge ohne Berücksichtigung der Termine zu ermitteln. Denn eine frühe Terminfixierung birgt die Gefahr, einzelne Vorgänge zu knapp bemessen oder sogar vollständig zu übersehen oder zu vergessen. Dies ist natürlich nicht immer möglich, z.B., wenn Abgabetermine bindend vorgegeben sind. In diesem Fall muss man sich arrangieren, beispielsweise, indem vereinbart wird, dass bestimmte Ergebnisdokumente weniger ausführlich ausfallen oder weniger wichtige Teile der Funktionalität zu einem weiteren etwas später liegenden Termin geliefert werden.

Bei der Erstellung eines neuen Softwaresystems oder der Veränderung eines bestehenden Systems fallen immer die gleichen Arbeitsschritte an, selbst, wenn sie stellenweise unterschiedlich benannt werden: Planung, Spezifikation der Anforderungen, Entwurf einer Lösung, Implementierung der Lösung, Test und Bewertung und schließlich Auslieferung und Inbetriebnahme. Ein Software-Entwicklungsprozess definiert für all diese Schritte nicht nur sinnvolle Detailabläufe, wie beispielsweise für die Durchführung von Reviews und Tests, sondern er definiert ebenso sämtliche Ergebnisse, die im Projekt pro Aufgabenstellung zu liefern sind. Häufig wird dabei zwischen internen und externen Lieferobjekten unterschieden. Somit haben der Projektleiter und alle Projektmitarbeiter eine vollständige Liste, anhand derer verifiziert werden kann, ob eine Aufgabe vollständig bewertet und bearbeitet wird. Es liegt nahe, die Daten über die Verteilung von Aufwand und Zeit möglichst vieler Projekte zu sammeln. So hat man die Möglichkeit, Vergleiche zu ähnlichen Projekten zu ziehen und aus der Erfahrung vergangener Projekte wertvolles Wissen für zukünftige Projekte abzuleiten.

DIE BEDEUTUNG DER ANFORDERUNGSSPEZIFIKATION

"Wer das erste Knopfloch verfehlt,
kommt mit dem Zuknöpfen nicht zu Rande."

- Goethe

Insbesondere bei komplexeren Projekten versuchen viele Auftragnehmer, Risiken zu minimieren, indem sie sich nicht auf Festpreisvereinbarungen einlassen, sondern nach Aufwand kalkulieren. Es wird auch argumentiert, schnell mit produktiver Arbeit beginnen zu wollen, um sich nicht zu lange mit Problemanalysen beschäftigen zu müssen. Allerdings wird damit die Sicherheit, ein großes Projekt zum Erfolg zu führen, keineswegs erhöht. Im Gegen-

satz zu Festpreisprojekten, bei denen der Auftragnehmer eine große Verpflichtung zur Einhaltung von Zeit, Inhalt und Kosten eingeht, beobachtet man bei Aufwandsprojekten regelmäßig, dass Termine und Aufwand bzw. Kosten aus dem Ruder laufen, was zur Unzufriedenheit beim Kunden führt.

In Wahrheit entscheidet jedoch nicht Festpreis- oder Aufwandscharakter über Erfolg oder Misserfolg eines Projekts, sondern viel mehr eine realistische und gleichermaßen für Auftragnehmer wie Auftraggeber verständliche Beschreibung der Anforderungen. P. Watzlawik wies auf die Bedeutung von Kenntnis und Verständnis der Ausgangssituation hin, indem er sagte: "Die Lösung ist das Problem". Je genauer und vollständiger die Anforderungen spezifiziert und verstanden sind, desto sicherer können Voraussagen über die Faktoren Zeit, Inhalt und Kosten gegeben und eingehalten werden und desto sicherer kann das weitere Vorgehen auf dieser Basis abgeleitet werden. Die Kunst im Zusammenspiel von Auftraggeber und Auftragnehmer besteht darin, herauszufinden, worin genau die Intention der Aufgabenstellung liegt und wie genau beide Parteien gewinnbringend zusammenwirken können.

Beinahe selbstredend ist, dass Festpreisvereinbarungen nur dort möglich sind, wo man sich auf vollständige und exakte Spezifikationen einlassen kann und will. Möglichkeiten zur Anpassung der Ziele, sprich der Änderung oder Erweiterung der zu Beginn beschriebenen Anforderungen, bleiben bei einer Festpreisvereinbarung dennoch gegeben, wenn diese durch ein abgestimmtes Verfahren für ChangeRequests (Änderungsanträge) ergänzt wird. Darüber hinaus ist eine Kombination von Festpreis- und Aufwandsvereinbarung möglich, wobei bestimmte Anteile nach Festpreis, andere auf Aufwandsbasis realisiert werden.

FAZIT

Wer erfolgreich große Softwareprojekte unter strengen kaufmännischen Rahmenbedingungen durchführen will, muss geeignete Prozesse entwickeln und betreiben. Das Wesen des Managements wäre unvollständig, wenn es den spezifizierten Entwicklungsprozess unter Nutzung neuester Werkzeuge und Technologien außer Acht ließe. Allerdings ist eine gehörige Vorleistung zu erbringen und es bedarf Know-how, um einen sinnvollen Entwicklungsprozess zu etablieren. Von besonderer Bedeutung ist es, alle Betroffenen und insbesondere die Mitarbeiter, bei der Entwicklung und schrittweisen Einführung des Prozesses aktiv zu beteiligen. Welcher Projektleiter würde es schon begrüßen, wenn seine Mitarbeiter sagen würden: "Schöne Bescherung. Unser Projektleiter hat ein Buch gelesen. Bisher haben wir gewusst, was er wollte, jetzt müssen wir es erraten."

Unsere Erfahrungen mit der Realisierung komplexer OO-Festpreisprojekte in den letzten zehn Jahren haben gezeigt, dass gelernte Abläufe, gelernte Ergebnistypen (spezifizierte Ziele) und eingespielte Arbeitsteilung die größten Potenziale für Effizienzsteigerungen enthalten. Mit modernster Technologie alleine werden keine Resultate erzielt, es bedarf ebenso angemessener Prozesse und Organisation.

Die Anforderungen an das Projektmanagement steigen mit Einführung eines Software-Entwicklungsprozesses. Der Projektmanager ist längst nicht mehr nur dafür verantwortlich, Aufgaben zu verteilen und deren Umsetzung zu kontrollieren. Er wird zusätzlich zum Prozessverantwortlichen für sein Projekt. Zu seinen Hauptaufgaben zählen jetzt Verhandlung und Kooperation mit seinen Mitarbeitern. Ebenso steigen die Anforderungen an die Projektmitarbeiter. Denn eigenverantwortliches Handeln setzt die nötige Bereitschaft und Kompetenz voraus. Diese muss

gelegentlich durch geeignete Ausbildungsmaßnahmen entwickelt werden. Reibungslose Kommunikation sowie Mittel und Bereitschaft zur Selbstkontrolle bilden das Fundament für Effizienz und Effektivität.

Der richtige Prozess lässt sich niemals aktenmäßig niederlegen und umgekehrt lässt er sich niemals auf diesem Wege erschaffen. Er muss sich kontinuierlich entwickeln, während er gelebt wird. Um echten Nutzen aus einem Software-Entwicklungsprozess zu ziehen, genügt es bei weitem nicht, die vielfach rezeptartig beschriebenen Praktiken mechanisch umzusetzen. Nur, wer sich um die zugrunde liegenden Prinzipien bemüht, die diese Praktiken motivieren, kann die erforderlichen Anpassungen an spezielle Rahmenbedingungen vornehmen.

LITERATUR:

- [1] Managing Software Requirements. A Unified Approach; Dean Leffingwell, Don Widring; Addison-Wesley 2000.
- [2] The Practice of Management; Peter F. Drucker; 1955. Deutsche Ausgabe: Die Praxis des Managements; Peter F. Drucker; Econ 1998, unveränderter Nachdruck von 1969
- [3] The Rational Unified Process. An Introduction; Phillipe Kruchten; Addison-Wesley 1998. Deutsche Ausgabe: Der Rational Unified Process. Eine Einführung; Addison-Wesley 1999.
- [4] Führen Leisten Leben; Fredmund Malik; DVA 2000
- [5] Das prozesszentrierte Unternehmen; Michael Hammer; Campus 1997
- [6] Abenteuer Kommunikation; Wolfgang Walker; Klett-Cotta 1998



Dipl.-Inform. Frank Ledermann leitet als Projektmanager Großprojekte bei der ARS NOVA Software GmbH. Parallel entwickelt er als Koordinator Software-Entwicklungsprozess den ARS NOVA Software Entwicklungsprozess kontinuierlich weiter und unterstützt Kunden bei der Entwicklung und Implementierung eigener Entwicklungsprozesse und -methoden. Email: frank.ledermann@arsnova.de