INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Prof. Dr. Dr. h.c. M. Broy
Dr. H. Ehler, S. Wagner

Lösungsblatt 1
26. Oktober 2006

Übungen zu Softwaretechnik

Aufgabe 1 – Erfolgsfaktoren der Softwareentwicklung

Für die erfolgreiche Durchführung von Softwareentwicklungsprojekten sind folgende Faktoren wichtig:

- *Unterstützung durch die Geschäftsführung*
- *Einbeziehung der Nutzer*
- *Motiviertes und kompetentes Team*
- *Erfahrene Projektleiter*
- *Verlässliche Schätzungen*
- *Eindeutige Geschäftsziele und Ownership*
- *Minimierung der Projektgröße*
- *Angemessenes Vorgehensmodell einschließlich Qualitätssicherung für den gesamten Software-Life-Cycle*
- *Standardisierte Software-Infrastruktur*
- *Klare und stabile Anforderungen*

Diese Stichpunkte für Erfolgsfaktoren sollen näher beschrieben und untersucht werden.

a) Konkretisieren Sie diese Stichpunkte.

- *Unterstützung durch die Geschäftsführung*

  Die Geschäftsführung muss einheitlich hinter den Entwicklungszielen und der Einführungsstrategie des Softwaresystems stehen. Entscheidend ist darüber hinaus, dass der Projektleiter durch schnelle und konsequente Entscheidungen bei Ressourcenzuteilung, Prioisierungen sowie bei der Lösung von Konflikten unterstützt wird. Nur so bekommt das Projekt den nötigen Rückhalt im Unternehmen.

- *Einbeziehung der Nutzer*

  Selbst wenn ein Projekt termin- und kostengerecht abgewickelt wird, ist es eine Fehlinvestition, wenn wesentliche Anforderungen der Nutzer nicht erfüllt werden. Deshalb ist es essenziell, die zukünftigen Nutzer eines Systems verantwortlich an Spezifikation, Abnahme und Benutzerorganisation zu beteiligen.

- *Motiviertes und kompetentes Team*

  Motivation verbunden mit Kompetenz sind die wichtigsten Produktivitätsmotoren. Es ist eine zentrale Aufgabe des Managements diese mit Hilfe einer produktivitätsfördernden Arbeitsumgebung und angemessener Führung sicher zu stellen.

- *Erfahrene Projektleiter*

  Der Projektleiter nimmt eine Schlüsselrolle für den Erfolg eines Software-Projekts ein. Er bildet die Schnittstelle zwischen Auftraggeber und Entwicklern und ist für den reibungslosen Ablauf des gesamten Projekts verantwortlich. Ein erfolgreicher Projektleiter benötigt deshalb neben der durchgängigen Projektverantwortlichkeit über alle Phasen des Projektes hinweg ein grundsätzliches Verständnis der Fachlichkeit, der Software-Technik sowie Erfahrung im Bereich der Organisation und der Menschenführung.

- *Verlässliche Schätzungen*

  Verlässliche Schätzungen von Aufwendungen und Nutzen eines Software-Projektes sind zum einen eine wesentliche Grundlage für die Entscheidung des Managements ob ein Projekt durchgeführt werden soll und zum anderen eine Voraussetzung für eine realistische Planung des Projektablaufes.

- *Eindeutige Geschäftsziele und Ownership*

  Die Geschäftsziele eines Software-Projektes müssen präzise festgelegt, kommuniziert und überprüft werden. Dabei ist besonders entscheidend, dass es in der Geschäftsführung eine klare Verantwortlichkeit gibt, für die Projektkosten sowie die Formulierung, Abstimmung und Umsetzung der Geschäftsziele inkl. des konkreten Nutzens.

- *Minimierung der Projektgröße*

  Die Größe eines Software-Projektes hat einen bedeutenden Einfluss auf dessen Erfolg. Je größer und komplexer ein Projekt ist, desto schwerer ist es zu beherrschen und desto wahrscheinlicher ist es, dass das Projekt scheitert. Nach [Stan99] ist die Erfolgswahrscheinlichkeit bei einem Projekt mit einem Personalaufwand größer als 3 Millionen Dollar höchstens 15% und bei über 10 Millionen Dollar ist sie nahezu Null – das Projekt wird, wenn überhaupt kaum ohne Termin und Kostenüberschreitungen sowie Abstriche bei der Qualität abgeschlossen werden können!

- *Angemessenes Vorgehensmodell einschließlich Qualitätssicherung für den gesamten Software-Life-Cycle*

  Ein auf die spezifischen Bedürfnisse eines Unternehmens angepasstes Vorgehensmodell für standardisierte Entwicklungs- und Wartungsprozesse zur Unterstützung der Entwickler einschließlich einer effektiven Qualitätssicherung etabliert sein. Die folgenden Aspekte müssen dabei insbesondere abgedeckt sein:

  - Minderung der Risiken durch ein Vorgehen in Phasen
  - Entlastung der Entwickler durch Vorgabe grundlegender Methoden, Werkzeuge und Standards
  - Sicherstellung der qualitativen Anforderungen wie der Einhaltung vereinbarter Standards oder der Konsistenz und Vollständigkeit der Phasenergebnisse durch konkrete Prozesse zur Qualitätssicherung

- *Standardisierte Software-Infrastruktur*

  Die funktionellen Anforderungen an ein Software-System ändern sich häufig schnell. Deshalb sollte zumindest die Software-Infrastruktur und -Architektur stabil sein, damit sich die Software-Entwickler auf ihre eigentliche Aufgabe konzentrieren können.

- *Stabilität der grundlegenden Anforderungen*

  Als erste Stufe eines Projekts wird ein minimaler Satz von grundlegenden Anforderungen festgelegt, der zentrale Funktionalitäten für das Unternehmen realisiert und als Basis für weitere Entwicklungsstufen dienen kann.

b) *Beschreiben Sie jeweils anhand eines Beispiels mögliche Auswirkungen bei Missachtung des Erfolgsfaktors.*

- *Unterstützung durch die Geschäftsführung*

  Der Vorstand eines Finanzdienstleisters hatte beschlossen, das seit 25 Jahren eingesetzte Informationssystem zur Unterstützung von Vertrieb und Produktverwaltung von Grund auf neu zu entwickeln.

  Finanzvorstand und Chief Information Officer (eingesetzt von dem auch für Informationsverarbeitung verantwortlichen Vorstand) bildeten den Projektlenkungsausschuss. Der Vorstandssprecher und Vertriebsvorstand waren nicht direkt in das Projekt eingebunden. Ihre Mitarbeiter lieferten allerdings viele Anforderungen an das zu entwickelnde System.

  Eine einheitliche Unterstützung des Projektes durch die Geschäftsführung war damit nicht gegeben. Von der Geschäftsführung gemeinsam akzeptierte und kommunizierte Projektziele existierten nicht.

  Jedes Mitglied der Geschäftsführung versuchte über seine Mitarbeiter Einfluss auf das Projekt zu nehmen. Die Projektziele veränderten sich dadurch permanent. Das Projekt wurde nach einer Überziehung des Gesamtbudgets um 600% abgebrochen.

- *Einbeziehung der Nutzer*

  In einem Software-Projekt mit einem Budget im zweistelligen Millionen-Euro-Bereich galt es, die Funktionalität einer neuen fachlichen Komponente zu spezifizieren. Das Team überzeugte die Projektleitung, Workshops mit den Anwendern des Systems zu veranstalten.

  Im Workshop wurde versucht, die Funktionalität und die Oberfläche der neuen Komponente zu erarbeiten. Früh im Workshop ergab sich die Frage, welche Produkte dem Kunden über das System angeboten werden sollten. Die Antwort des „Anwenders“: „Das kann ich Ihnen aus dem Stegreif nicht sagen, aber vor einigen Monaten habe ich das bereits einem Programmierer im zweiten Stock erzählt. Der müsste es wissen.“

  Nach einigem Hin und Her stellte sich heraus, dass die vermeintlichen Anwender nicht wirklich die Nutzer des Systems waren, sondern im Unternehmen nur die Schnittstelle zu den Kunden des Unternehmens, den tatsächlichen Anwendern, repräsentierten. Neben ihren eigentlichen Aufgaben hatte man ihnen deshalb noch zusätzlich die Spezifikation der Anforderungen an die Software übertragen. Sie waren also nur ein Puffer zwischen den Nutzern und dem Entwicklerteam. Dabei gingen viele Anforderungen verloren, wurden verfälscht oder neu erfunden.

  Nach Einführung des Gesamtsystems mussten über die Hälfte der Masken nochmals angepasst werden. Dies führte zu über 5 Millionen Euro zusätzlichen Entwicklungskosten.

- *Motiviertes und kompetentes Team*

  Ein Finanzdienstleister entschied sich dafür, ein neues Internet-basiertes System zu entwickeln. Er beauftragte dafür im Rahmen eines Dienstleistungsvertrags einen international tätigen Softwaredienstleister. Der

Auftragnehmer startete mit einem kleinen Team von Experten, vergrößerte jedoch das Team rasch und zog für die Implementierung überwiegend neu eingestellte Anfänger heran. Der Auftraggeber stellte die Arbeitsumgebung bereit. Aufgrund der Raumknappheit saßen bald über 40 Entwickler über Monate hinweg in zwei mittelgroßen Räumen – die meisten mussten sich dabei einen Schreibtisch mit einem oder sogar zwei weiteren Entwicklern teilen.

Die Qualität des entstandenen Systems war insgesamt schlecht; viele Implementierungsfehler führten dazu, dass die Kernfunktionalität erst drei bis sechs Monate nach dem geplanten Meilenstein ausgeliefert werden konnte.

- *Erfahrene Projektleiter*

  In einem Software-Projekt mit einem Volumen von mehreren Millionen Euro wurde an Fachkonzept und Architekturentwurf gearbeitet. Die Voraussetzungen waren günstig: Ein gutes Pflichtenheft lag bereits vor, erste Pläne und Teilergebnisse wurden schnell und in hoher Qualität erarbeitet, die Kommunikation mit dem Kunden gestaltete sich angenehm und kooperativ.

  In dieser viel versprechenden Situation schaffte es der Projektleiter, der noch nie an einem Software-Projekt dieser Art beteiligt war, quasi im Alleingang, das Projekt zum Scheitern zu bringen. Durch das Zurückhalten von Informationen zwischen Geschäftsleitung, Anwendern und Entwicklern brachte er zunächst die Kommunikation im Projekt zum Erlahmen. Als sich dadurch erste Komplikationen ergaben, schlug er sich auf die Seite des Auftraggebers und übertrug die unvertrauten und lästigen Aufgaben bei Planung und Controlling sowie die Leitung der Entwicklung vollständig an einen externen Mitarbeiter, den er vor der Geschäftsleitung, Auftraggebern und Anwendern abschottete.

  Der Auftraggeber entzog dem Auftragnehmer schließlich auf Grund von mangelndem Vertrauen das Projekt und übertrug es einem anderen Unternehmen. Insgesamt ergab sich für das Projekt eine Verzögerung von etwa einem Jahr und ein zusätzlicher Aufwand von etwa 3 Millionen Euro.

- *Verlässliche Schätzungen*

  In dem im ersten Punkt beschriebenen Projekt wurde statt einer, von fachlicher und technischer Seite bestätigter Wirtschaftlichkeitsberechnung, bestehend aus erwarteten Kosten und dem zu erzielenden Nutzen des Projektes nur eine grobe Schätzung der Entwicklungskosten durch den CIO als Basis für den Start des Projektes akzeptiert.

  Im Verlauf des Projekts stiegen die Kosten von geschätzten 7,5 MIO € auf 40 MIO € für 80% der Funktionalität. Das wurde Projekt eingestellt, insbesondere weil schon die Betriebskosten der neuen Anwendung höher als der Nutzen geschätzt wurden.

- *Eindeutige Geschäftsziele und Ownership*

  Die Entwicklungs- und Betriebskosten des neuen Gepäckabwicklungssystems eines internationalen Flughafens waren so hoch, dass sich die Investition erst nach 1000 Jahren amortisiert hätte [Glas98, Flow96]. Das heißt, die alte Gepäckabwicklung hätte noch 1000 Jahre zu denselben Kosten betrieben werden können.

  Das primäre Geschäftsziel des Projektes war aber Kostenersparnis durch Rationalisierung. Das Projekt wäre in dieser Art nie gestartet oder zumindest frühzeitig gestoppt worden, wenn dieses Geschäftsziel von den Verantwortlichen kommuniziert und überprüft worden wäre.

- *Minimierung der Projektgröße*

  Eine Kooperation von drei Finanzdienstleistern schrieb ein Festpreisprojekt zur Neuentwicklung eines umfangreichen Abwicklungssystems aus. Die Ausgangsbasis war ein erster Anforderungskatalog von etwa 25 Seiten. Der Auftrag wurde an den billigsten Anbieter, ein weltweit agierendes Software-Haus, vergeben.

  Unter der Regie des Auftragnehmers wurde daraufhin ein umfassendes Fachkonzept entwickelt, das die Wünsche aller drei Auftraggeber repräsentierte. Aufgrund dessen war es extrem komplex. Erst nach etwa drei Jahren Laufzeit mit einem großen Entwicklungsteam war der erste Implementierungsmeilenstein testbar. Die Test-Ergebnisse waren bzgl. Funktionalität und Performance inakzeptabel. Das Projekt wurde abgebrochen, da die Hochrechnung der ursprünglichen Projektkostenschätzungen einen Faktor vier ergab. Der gesamte Schaden betrug mehr als 50 Millionen Euro.

- *Angemessenes Vorgehensmodell einschließlich Qualitätssicherung für den gesamten Software-Life-Cycle*

  In einem großen Entwicklungsprojekt war das anfänglich kleine Entwicklerteam von weniger als fünf Personen auf insgesamt über 50 Entwickler angewachsen. Während der gesamten Entwicklung wurde angestrebt möglichst schnell umfangreiche neue Funktionalität zu entwickeln. Für die Bereitstellung einer angemessenen Verfahrenstechnik standen nur marginale Mittel und Ressourcen zur Verfügung.

  Nach insgesamt fünf Jahren Arbeit ohne ein dokumentiertes Vorgehensmodell mit entsprechenden Standards und Qualitätskontrollen und mit ungenügend integrierten Werkzeugen war neben dem Quellcode eine unüberschaubare Vielzahl an uneinheitlich strukturierten Modellen und Dokumenten entstanden (Dateien im Datei- oder Versionsverwaltungssystem, Daten in Datenbanksystemen, E-Mails, Ausdrucke, etc). Dies führte dazu, dass die Kosten für die Weiterentwicklung und Wartung des Systems untragbar wurden: Entwickler konnten erforderliche Informationen nicht oder nur nach stundenlangem Suchen finden und weigerten sich, bestehenden Code zu ändern oder anzupassen, da sich bei jeder Änderung unvorhergesehene Effekte ergeben konnten.

  Die Projektleitung entschloss sich, ein angemessenes Vorgehensmodell zu definieren und in diesem Rahmen auch die Dokumentation neu zu strukturieren. Alleine die Sammlung, Kategorisierung und Neuablage der relevanten Dateien und Dokumente (insgesamt über 20.000, bei einer anfänglichen Schätzung von 1.500) kostete mehr als 3 Personenjahre. Die zusätzlich erforderliche Zeit für eine inhaltliche Aufarbeitung und Konsolidierung wurde von den Bearbeitern auf 30 bis 100 Personenjahre geschätzt, ebenso der bereits während der Entwicklung aufgelaufene Produktivitätsverlust.

- *Standardisierte Software-Infrastruktur*

  Bei einem Dienstleistungsunternehmen mit einem großen Filialnetz wurden die unterschiedlichen Anwendungssysteme zur Sachbearbeitung für verschiedene Fachgebiete ohne eine standardisierte Software-Infrastruktur entwickelt.

  Nach etwa 6 Jahren stand eine Reihe von Anwendungen zur Verfügung. Es existierte aber unter anderem kein einheitlicher Style-Guide für Benutzeroberfläche einschließlich Help-Funktion, es gab kein einheitliches Rechtekonzept mit Single-Logon und zentrale Steuerungsdaten waren

redundant auf 54 unabhängige Tabellen verteilt. Die Anwendungssysteme waren folglich umständlich zu bedienen sowie schwer zu warten oder zu erweitern.

Die Vereinheitlichung der Benutzeroberflächen einschließlich Single-Logon-Verfahren kostete etwa 30 Personenjahre. Die Konsolidierung der Steuerungsdaten kostete etwa 5 weitere Personenjahre und eine einheitliche Hilfefunktion nochmals etwa 5 Personenjahre.

- *Klare und stabile Anforderungen*

  In dem im ersten Punkt beschriebenen Projekt wurde entschieden, die gewünschte Funktionalität parallel für alle Produkte zu entwickeln. Dies führte dazu, dass bis zum ersten testbaren Software-System drei Jahre vergangen sind, in denen die Entwickler gegen eine, sich ständig ändernde Systemarchitektur entwickeln mussten – dies deshalb, da zusätzlich gegen das Prinzip der sequentiellen Abwicklung der Phasen verstoßen wurde und ständig während Programmierung das Objektmodell einschließlich Datenmodell geändert wurde.

  Besser wäre es gewesen zunächst das Objektmodell an Hand eines Produktes stabil zu entwickeln und zu testen um dann in weiteren Teilprojekten die restlichen Produkte zu ergänzen. Zusätzlich wären dadurch erste Ergebnisse früher nutzbar gewesen.

c) *Ordnen Sie nach Ihrer Einschätzung der Wichtigkeit den Erfolgsfaktoren ein Gewicht für den Anteil am Projekterfolg in Prozent zu.*

Die Standish Group führt seit 1994 etwa alle zwei Jahre Umfragen beim Management von Unternehmen nach den wesentlichen Erfolgsfaktoren für Software-Projekte durch. Die Ergebnisse werden jeweils in einem Bericht an den amerikanischen Präsidenten veröffentlicht. In [Stan01] wurden folgende zehn Erfolgsfaktoren genannt und für deren Einfluss auf den Erfolg eines Projektes folgende Werte angegeben (in Klammern jeweils die Werte von 1999 bzw. 1995):

1) Unterstützung durch die Geschäftsführung: 18% (15%, 14%)
2) Einbeziehung der Nutzer: 16% (20%, 16%)
3) Erfahrene Projektleiter: 14% (15%, -)
4) Eindeutige Geschäftsziele und Ownership: 12% (15%, 8%)
5) Minimierung der Projektgröße (Meilensteine in kurzen Abständen): 10% (10%, 8%)
6) Standardisierte Software-Infrastruktur: 8% (0%, 0%)
7) Klare und stabile Anforderungen: 6% (5%, 13%)
8) Angemessenes Vorgehensmodell einschließlich Qualitätssicherung für den gesamten Software-Life-Cycle: 6% (0%, 0%)
9) Verlässliche Schätzungen: 5% (5%, 10%)
10) Motiviertes und kompetentes Team: 5% (5%, 7%)

[Flow96] S. Flowers. Software Failure: Management Failure. John Wiley & Sons. 1996.

[Glas98] R. Glass. Software Runaways. Prentice Hall. 1998.

[Stan01] Standish Group International, Inc. Collaborating on Project Success. Software Magazine, February/March 2001. Wiesner Publishing. 2001.