

Auswirkungen der Qualitätssicherung in Projekten

Martin Wenger

Zusammenfassung

Dieser Erfahrungsbericht zeigt die Rolle der Qualität in der Softwareentwicklung eines komplexen Grossprojektes im Finanzbereich. Aus Sicht der Qualitätssicherung wird der Projektverlauf dargestellt.

Die Analyse des Geschäftsprozesses im Retail Office deckte hohes Optimierpotenzial in den bankinternen Abläufe auf, dessen Verbesserung direkten Nutzen für die Bankkunden und Kundenberater darstellt. Mit der zu entwickelnden IT-Lösung wird der Geschäftsprozess abgebildet und automatisiert. Die Anbindung an ein bestehendes, historisch gewachsenes Hostsystem musste geschaffen und neue Technologien in einem Rechenzentrum aufgebaut und eingeführt werden. Der Paradigmenwechsel von der Hostentwicklung zu modernen Methoden des Software Engineerings wurde vollzogen, was mit einem grossen Veränderungsaufwand verbunden war.

Mit Hilfe des Modells des magischen Quadrates werden der Projektverlauf und die Zusammenhänge zwischen den Dimensionen Kosten, Termin, Funktionsumfang und Qualität erklärt. Im Verlauf des Projektes verändern sich die Parameter zu ungunsten der Qualität. Die Softwareproduzenten müssen lernen, mit mangelnder Qualität umzugehen und diese beheben. Als möglicher Ansatz kann die Qualitätssicherung durch eine externe, neutrale Instanz wahrgenommen werden. Bei der Qualitätsmessung sind einfache Metriken einzusetzen, damit diese in einer heterogenen Teamzusammenstellung von allen verstanden und vom Qualitätsverantwortlichen kommuniziert werden können.

Die Qualitätssicherung führt oft zu zwischenmenschlichen Konflikten, die zusätzlich bewältigt werden müssen. Der Nutzen der Qualitätssicherung muss allen Betroffenen im Projekt aufgezeigt werden: Das Entwicklungsteam profitiert von der Sicherheit eines 'second looks' und wird von der Testarbeit befreit. Für den Softwareproduzenten eröffnen sich neben den finanziellen Einsparungen, die Chance die Kunden- und Marktbedürfnisse besser zu treffen, da Fehler bereits in der Entwicklungsphase erkannt werden können.

1. Einleitung

Die Real-Time Center AG (RTC) ist ein Informatik-Dienstleistungsunternehmen, das mit der Standardsoftware IBIS® und einem umfassenden Angebot sämtliche Dienstleistungen abdeckt, welche eine Bank für das Outsourcing der Informatik benötigt. Sie bietet ein Produktportfolio für den Finanz-Sektor an, angefangen bei der Software-Entwicklung über den Rechenzentrumsbetrieb bis zu Produkten und Dienstleistungen im dezentralen Bereich. Im dezentralen Bereich der Clients werden die Banken mit umfassender Unterstützung versorgt, die über Software-Installation und -Verteilung, Netzwerkbetrieb, Versions- und Konfigurationsmanagement, Userverwaltung, Serverbetrieb, Support, Schulung die ganze Bandbreite der Informatikdienstleistungen abdeckt.

Die Rechenzentrums- und Outsourcingfunktionen werden von sechs Banken (BEKB, BLBK, BSKB, BCJ, AKB, und Migros) und weiteren Dienstleistern im Finanzbereich (Coop, RBA, SwissBankers, Atag Asset Management) genutzt.

2. Ausgangslage

2.1. Problematik aus Endkundensicht

Bei den Bankfunktionen im Retail Office (Konto eröffnen, mutieren u.a.m.) und der Auskunftsbereitschaft der Kundenberater wurde erkannt, dass die Abläufe im Front-Office teuer, arbeitsintensiv, nicht kundenfreundlich und fehlerbehaftet sind.

Das Grundproblem lag darin, dass am Frontdesk Formulare und Papiere zuhanden Backoffice erstellt wurden und im Gesamtablauf nicht schnell und effizient zur Befriedigung des Kundenbedarfs abgewickelt wurden. Die Kundenorientierung und Steigerung der Auskunftsbereitschaft der Banken steht im Vordergrund, daher muss der Geschäftsablauf so verbessert werden, dass

- der Kunde unmittelbar bedient wird (nach Abschluss des Kundengesprächs sind alle administrativen Tätigkeiten erledigt und die Transaktion ist beim Verlassen der Bank abgeschlossen) und somit die Durchlaufzeit in der Bank verkürzt wird,
- alle Daten werden nur noch einmal erfasst, und
- die Erfassung der Daten so einfach und standardisiert ist, dass die Qualität erhöht wird.

2.2. Vorgehen

Zur Lösung dieser Problematik wurden als erstes die Geschäftsprozesse analysiert und ein Konzept zur Verbesserung der Situation erarbeitet. Die vorgeschlagene Prozessveränderung wurde dann mit organisatorischen Massnahmen auf die Gebrauchstauglichkeit geprüft. Anschliessend wurde zur Automatisierung und Unterstützung des Geschäftsprozesses durch Datenverarbeitungsmittel die Entwicklung von KUBA („Kundenberaterapplikation“) in Auftrag gegeben. Die sechs Banken, als Hauptabnehmer der Rechenzentrumsfunktionen, haben sich zusammengeschlossen und bilden gemeinsam die Trägerschaft für das Projekt KUBA. Eine der Banken wurde als treibende fachliche Kraft (Lead-Bank) nominiert, die als erste Bank mit der neuen Lösung ausgerüstet werden soll.

2.2.1. Zielsetzung der Kundenberaterapplikation

Die Ziele des Systems Kundenberaterapplikation 'KUBA' sind:

- Einmalige Datenerfassung am Kundenberater-Arbeitsplatz, schnelle und einfache Prozesse
- Drastische Reduktion der Anzahl Kundenformulare durch den Einsatz von Standardtexten
- Hohe Auskunftsbereitschaft (Kundeninformationssystem)
- Aufbau auf bestehender Infrastruktur
- Standardisierung der Abläufe
- Beschleunigung der Kernprozesse für den Kundenberater (weniger Nacharbeit)
- Erledigung des Geschäfts kurz (max. 10 Minuten) nach Kundenberatung
- Erhöhte Datenqualität

KUBA bildet den generellen Ablauf aller Banken als gemeinsame Basis ab, mit der Möglichkeit die individuellen Besonderheiten der Banken flexibel anzupassen, d.h. die gleiche mandantenfähige Software, die im Rechenzentrum einheitlich betrieben wird, kann mit unterschiedlichen aussehenden Oberflächen und Abläufen der einzelnen Banken konfiguriert werden.

2.3. Vorgaben und Rahmenbedingungen

2.3.1. Anbindung von bestehenden Systemen

Zur Vermeidung von spezialisierten Punkt-zu-Punkt Verbindungen wurde ein Kommunikationsbus (Tuxedo) ausgewählt und aufgebaut. Die folgende Abbildung zeigt, wie

beispielsweise bei 5 Applikationen bis zu $n \cdot (n-1) / 2 = 10$ Punkt-zu-Punkt Verbindungen entstehen können.

Abbildung: Kommunikationsbus

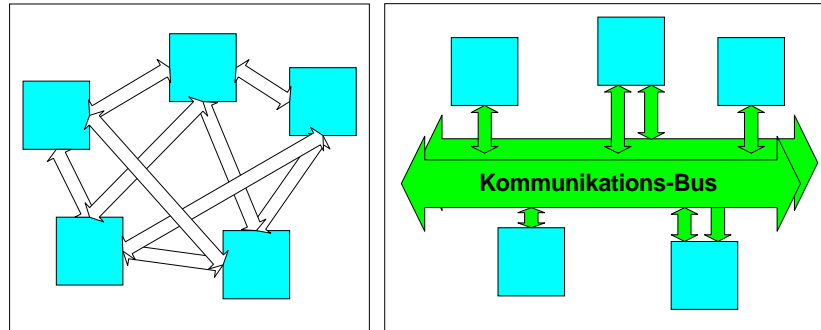


Tabelle: Bestehende Systeme

IBIS-Basis-System (Host)	umfassende bankfachliche Applikation, basiert auf Transaktionen für verschiedene Oberflächen und Anwendungen
IBIS-PFS	Personen-Führungs-System (wie Geschäftspartner u.a.m.)
IBIS-UPS	Unterschriften Prüfsystem

2.3.2. Einbezug der neuen Architektur

Die RTC als Outsourcing Partner hat zudem Bedarf die IT-Architektur zu erneuern und daher ein Architekturprojekt gestartet. Als Voraussetzung für die Entwicklung von KUBA muss die neue Architektur verwendet werden. Das Architekturprojekt wurde vor dem Applikationsprojekt gestartet und musste zu einer Einheit integriert werden. Die neue Architektur basiert auf der n-tier Aufteilung. Da bisher noch keine Erfahrungen im Bereich der Performance, der idealen Verteilung und Konfiguration des Systems vorlag, wurden alle möglichen Varianten der Architektur zur Evaluation und Test offen belassen.

Abbildung: Varianten n-tier

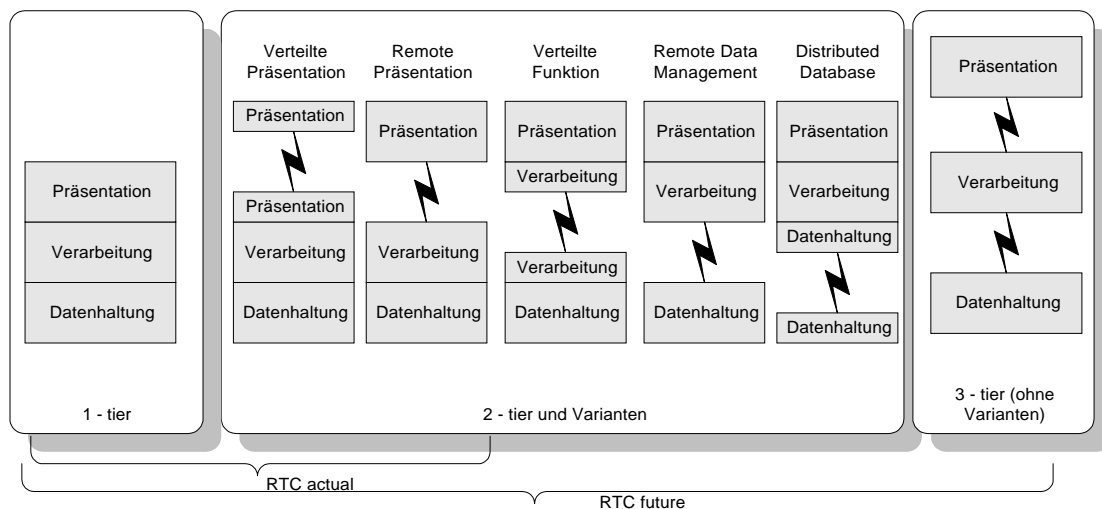


Tabelle: n-tier Architektur

Beschreibung	Bewertung
1 tier Monolithische Applikation	Vorteil der geringeren Komplexität. Nachteil der schlechten Zugänglichkeit von Daten, der mangelnden Flexibilität.
2 tier Zweiteilung, wobei viele Varianten (siehe Abbildung) möglich sind.	Häufig verwendetes Modell bei datenbankzentrierten Anwendungen. Potentielle Nachteile für Flexibilität und Skalierbarkeit.
3 tier Alle drei Ebenen können voneinander getrennt werden.	Vorteil der grossen Flexibilität, der spezialisierten Ausnutzung von Ressourcen (Rechnern) und der einfacheren Erweiterung. Nachteil der Komplexität.

Der Einsatz des Kommunikationsbuses und der 3-tier Architektur, dienen im Projekt zur Entkopplung und Anbindung der Altsysteme und wird als Basisarchitektur für alle weiteren Projekte eingesetzt.

2.4. Teambzusammensetzung und erforderliches Wissen

Zur Bewältigung der vielfältigen Aufgaben, der hohen Anforderung und dem breit gestreuten Wissens musste ein heterogenes Team aufgebaut werden mit

- Mitarbeitern aus mehr als zwölf internen Abteilungen,
- fachlichen Vertretern aus den sechs Banken, sowie
- mehr als fünf externen Beratern.

Abbildung: Rollen und Kenntnisse

Rollen	Erforderliche Kenntnisse
Applikationsbenutzer Kundenberater Mutationszentrale 1st-level-supporter Benutzer-Supporter und Schulung 2nd, 3rd-level-supporter Schulung (KUBA, Betrieb, ...) Entwickler (Host, Middleware, Clients) Analyst Designer Programmierer Tester QS-Management Qualitätsverantwortlicher, Testing Betreiber Applikations-Betreiber (IBIS, PFS, UPS, ...) Technik-Betreiber (Host, Client/Server, Unix, NT, Oracle,..) Systembetreiber (Installation, Verteilung, Security, ..) Netzwerk IT-Architekt Projektleitung technisch fachlich Geschäftsleitung Bankenvertreter Consultants Entwickler IT-Architekten Middleware (Tuxedo) Qualitätssicherer Auditoren	<ul style="list-style-type: none"> • Business-Prozesse • Existierende technische Komponenten und Hardware • Vorhandene Plattformen/Betriebssysteme (Unix, NT) • Vorhandene Applikationen • GUI-Tool • GUI-Plattformen (SAL 2, Browser) • Server-Plattformen (NT, Unix) • IT-Management • Grundlagen objektorientierte Methoden • GUI-Tool-Sprache • Applikationssprache (JAVA) • System-Programmiersprache (C/C++) • Kommunikationsbus (TUXEDO, SGL*Net, RMI) • Datenbanken (Oracle) • Host-Programmiersprachen (COBOL, C) • Host-Datenbanken (DMS, RDMS) • Host-Tools (z.B. PROPIP) • Existierende Host-Applikationen (TPs) • Middleware (TUXEDO, RMI) • Security • Test-Scriptsprachen (tcl, jacl) • Test-Tools • Datenbanktechnologien • Transaktionsmonitore • Netzwerke/Kommunikation

2.5. Chronologie des Projektverlaufes

Die Vorstudie zum Projekt wurde Ende 1997 abgeschlossen. Als Resultat dieser Arbeiten entstanden die Kundenanforderungen für die Kundenberaterapplikation (KUBA), die als Grundlage für alle weiteren Arbeiten dienten.

Bis Juni 1998 wurde in enger Zusammenarbeit mit den Banken ein visueller Prototyp (mit Masken) mit der Methode RAD (rapid application development) entwickelt. Alle nötigen Grundlegendokumente (wie Schnittstellen zu den Umsystemen, einzelne Designdokumente, Regeln, Abläufe, Abhängigkeiten u.a.m.) konnten erarbeitet werden.

Ein Jahr nach Projektstart wurde die Firma APP Unternehmensberatung AG (APP; www.APP.ch) mit der Aufgabe der externen Qualitätssicherung beauftragt und hat im Herbst 1998 die projektbegleitende Qualitätssicherung aufgebaut und institutionalisiert. Die Ergebnisse der Vorbereitungsarbeiten entsprachen der üblichen Praxis und den Theorien der Qualitätssicherung [1][2][7] (QS-Plan, Prüfplan, Testarchitekturen). Die Mitarbeiter der bestehenden hostlastigen Qualitätsabnahme wurden in den Techniken der projektbegleitenden Qualitätssicherung ausgebildet.

3. Die Rolle der Qualität

3.1. Das magische Quadrat

Was passiert im Prozess der Softwareentwicklung (Softwareproduktion)? [4][8]

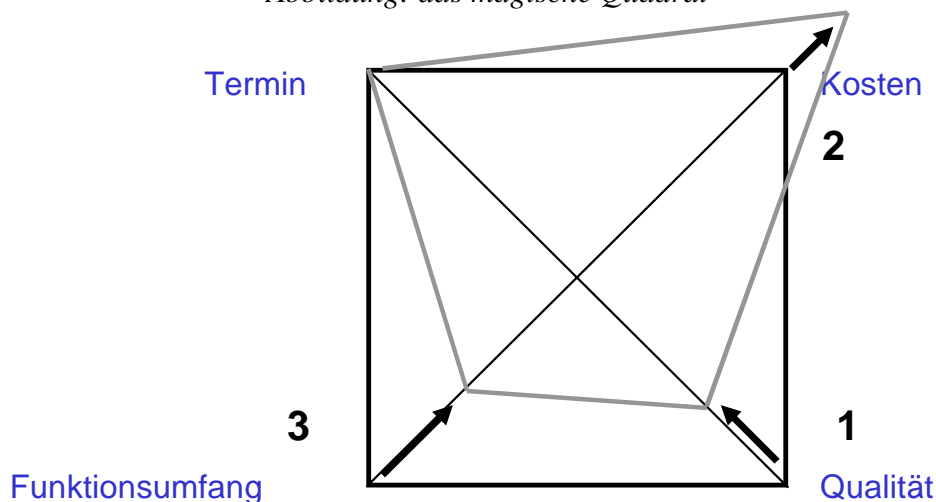
Die Kosten und Termine für ein Projekt werden durch das Management oder den Auftraggeber festgelegt. Diese Parameter sind normalerweise fix. Mit fortschreitender Projektdauer wird die Zeit knapp und die Entwicklung der Funktionen ist nicht abgeschlossen.

1) Die Termine müssen eingehalten werden, wobei der Druck im Projekt erhöht wird. Dadurch schleichen sich Fehler ein oder im Projekt wird versucht, das Vorgehensmodell zu umgehen (Abkürzungen ausserhalb der Planung). Dies führt dazu, dass die Qualität sinkt und die Fehlerzahl zunimmt.

2) Mit sinkender Qualität steigen die Kosten für die Fehlerbehebung wegen der Mehrarbeit.

3) Damit die Termine trotzdem eingehalten werden können, wird der Funktionsumfang verringert ('das liefern wir eben in einem späteren Release').

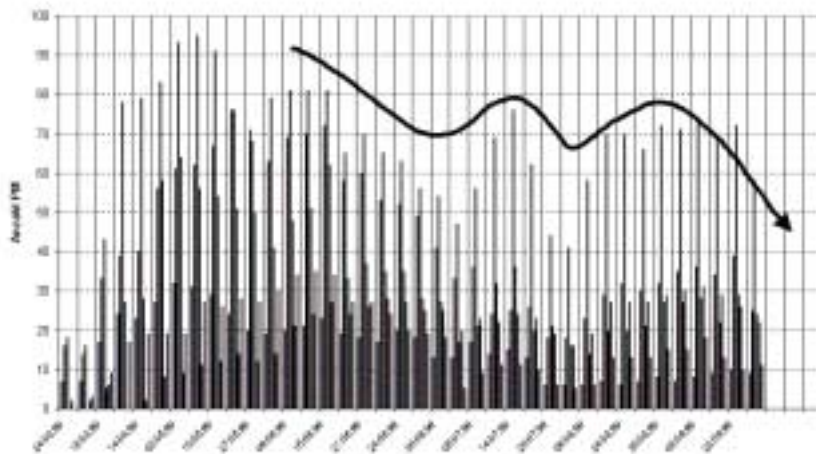
Abbildung: das magische Quadrat



3.2. Fehlerverlauf im Projekt

Die Anzahl der gefundenen Fehler in KUBA beläuft sich nach 8 Monaten auf tausend. Dies ist eine absolut normale Situation in der Softwareproduktion [4]. Die Qualitätssicherung im Projekt KUBA funktioniert, denn jeden Fehler, den das Entwicklungsteam findet und beheben kann, findet nicht der Kunde.

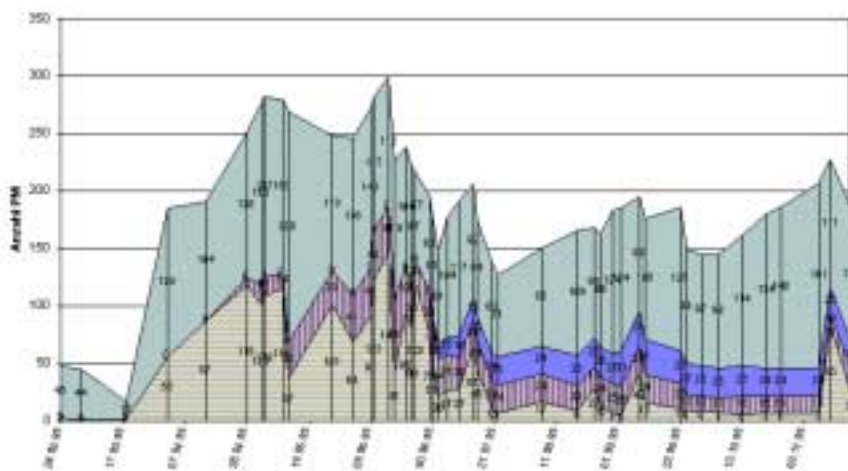
Abbildung: Verlauf der Problemmeldungen gesamt



Fehlerklassierung:

- Fatale Fehler
- Schwere Fehler
- Andere Fehler
- Kleine Fehler
- Fragen
- Änderung

Abbildung: Verlauf offene Problemmeldungen



Möglicher Status der Problemmeldungen:

- In Bearbeitung
- Pendent
- Zurückweisen
- In Test

Die Fehler aller Tests (HW/SW-Integration, Systemgenerierung, Funktion, Leistung, Degradation, Datensicherheit) wurden als Problemmeldung erfasst, klassiert und periodisch mittels obigen Statistiken ausgewertet. Die Problemmeldungen als gemeinsame Komponente der Kommunikation im Team haben sich bewährt. Sie bergen aber Konfliktpotenzial bei streitigen oder nicht genau zuteilbaren Fällen.

Zur unmissverständlichen Kommunikation der Fehler und Problemmeldungen in diesem heterogenen Team haben sich die einfache Art der Fehlerklassierung –ohne die Verwendung von allzu theoretischen Modellen und Methoden [9]– bewährt.

Quintessenz: Bei der Qualitätsmessung sind einfache Metriken einzusetzen, damit diese in einer heterogenen Teamzusammenstellung von allen verstanden und vom Qualitätsverantwortlichen kommuniziert werden können.

Der Durchlauf der Fehlerbehebung war enorm. Die Dauer der Entwicklungs- und Testzyklen betrug 8 Monate. Dies ergibt eine Fehlerbehebungsrate von 5.6 Fehler pro Arbeitstag.

Tabelle: Rate der Fehlerbehebung

Dauer der Qualitätssicherung	8 Monate = 240 Tage = 160 Arbeitstage
Anzahl Fehler	1000
Behobene Fehler	900
Fehlerbehebung	5.6 Fehler / Arbeitstag

Ist es möglich, die hohe Fehleranzahl zu minimieren? Die oberste Regel lautet, Fehler nicht zu machen, zumindest den gleichen Fehler nur einmal zu produzieren [3]. Es ist erwiesenermassen unmöglich, eine fehlerfreie Software zu schreiben [1].

Nicht die Anwender von Software-Produkten müssen lernen mit Fehlern zu leben, sondern die Softwareproduzenten.

3.3. Qualitätssicherung - ein Ansatz

Wie können Softwareproduzenten die Qualität ihrer Softwareprodukte in den Griff bekommen? Eine mögliche Lösung ist eine externe, neutrale Qualitätssicherungsstelle einzusetzen, welche die Aufgabe hat, die Qualität im Informatikgrossprojekt zu prüfen und zu sichern.

Die konkreten Aufgaben der externen Qualitätsberater sind

- die Ergebnisse in einem 'second look' zu prüfen und
- im Sinne von einer 'second opinion' zu qualifizieren.

Der Softwareproduzent erwartet eine umfassende Prüfung seines Projektes und seiner Ergebnisse. Daher muss der Qualitätsberater Probleme erkennen und die nötigen Massnahmen vorschlagen. Er versucht, die Softwareentwickler umfassend bei der Erreichung der Ziele zu unterstützen.

3.3.1. Vorgehen der Qualitätssicherung

Die Abwicklung der Qualitätssicherung lässt sich in die folgenden zwei Phasen gliedern:

- **Aufbau** der qualitätssichernden Massnahmen in enger Zusammenarbeit mit der Projektleitung, damit die Ziele des Projekts und der Qualitätsprüfung definiert und das Vorgehen bestimmt werden können. In dieser Phase ergibt sich eine Bindung und gegenseitiges Vertrauen zwischen dem Projektleiter und dem Qualitätsberater. Der Berater wird zum Kollegen. Dies hat zur Folge, dass die geforderte Neutralität des externen Qualitätsberaters nur noch bedingt gegeben ist. Diese Phase ist eine Gratwanderung zwischen „Vertrauen kontra Unabhängigkeit“.
- **Durchführung** der qualitätssichernden Massnahmen, das heisst das Messen der Qualität mit geeigneten Mitteln wie Tests [6], Reviews, Audits usw. [1][2]. Die Messung der Qualität und deren Auswertung zeigt in dieser Phase, dass die Qualität ungenügend ist. Zudem nehmen die Unstimmigkeiten im Projekt und Spannungen im Team zu [13]. Der Qualitätsberater steht plötzlich in der Rolle des Polizisten oder des Sündenbocks. Die Beziehung zwischen dem Berater und dem Projektleiter kühlt sich ab, da die

Qualitätssicherer, wenn sie ihre Arbeit richtig machen, schonungslos Fehler und Versäumnisse aufdecken.

Quintessenz: Es besteht eine Zwangssymbiose, denn ohne Qualitätssicherung ist die Qualität nicht gegeben und ohne Qualität ist das Erreichen der Ziele unmöglich.

Das Konfliktpotential und die 'Softfactors' treten bei jeder Teamarbeit auf, müssen jedoch bei den qualitätssichernden Arbeiten besonders beachtet werden. Wenn an einer Entwicklungsanstrengung mehrere Parteien beteiligt sind, sind Interessenskonflikte unvermeidlich. Der Bereich der Systementwicklung und -installation ist für Konflikte besonders anfällig [13].

3.3.2. Nutzen der Qualitätssicherung

Wichtig ist es, den Nutzen der Qualitätssicherung allen Betroffenen im Umfeld des Projektes zu kommunizieren, denn

- das Entwicklungsteam profitiert von der Sicherheit eines 'second looks' und der Gewissheit, dass andere Kollegen die Arbeit nochmals prüfen, bevor der Kunde unzufrieden sein könnte. Ein Zusatznutzen für die Entwickler ist die Entlastung von der 'lästigen' Testarbeit, wobei gemäss Vorgehensmethodik die Einzeltests in jedem Fall Sache der Entwickler bleibt,
- für den Softwareproduzenten und die gesamte Unternehmung eröffnen sich neben den finanziellen Einsparungen, die Chance die Kunden- und Marktbedürfnisse besser zu treffen, da Fehler bereits in der Entwicklungsphase erkannt werden können ('second opinion'),
- die Endanwender werden es dem Softwareproduzenten danken, ebenso die Verantwortlichen der Schulung und des Supports. In der Softwareindustrie gilt leider zu oft noch die Bananentaktik: „Grün ernten und beim Kunden reifen lassen“.

3.3.3. Chancen

Das Projekt KUBA hat beim Softwareproduzenten Veränderungen ausgelöst, die zu bewältigen waren. Die neue Architektur und der Paradigmenwechsel von der Hostentwicklung zu modernen Methoden des Software Engineerings musste vollzogen werden. Die Applikation KUBA ist nicht mehr transaktionsorientiert und schliesst den Geschäftsprozess umfassend ein. Die Kultur der offenen Kommunikation und Fehlertoleranz, sowie die enge Zusammenarbeit mit externen Beratern und den vielen Mitarbeiter/innen im heterogenen Team musste erkannt, eingeschliffen und gelernt werden [12][13].

Jede Veränderung, gleichgültig ob sie positiv oder negativ wahrgenommen wird und ob sie selbst- oder fremdverursacht ist, bedeutet für die betroffenen Menschen Verlust von Vertrautheit, Sicherheit und Kontrolle [10]. Ihre Bewältigung erfordert geistige und seelische Energie und ist gewöhnlich mit beruflichen und privaten Risiken verbunden. Die Aspekte des 'Change Managements' gewannen an Bedeutung [11][14].

Der Vorwurf, dass man einiges noch nie so gelöst hat und dass das 'Neue' schlecht oder nicht umsetzbar sei, war im Projekt KUBA oft vorhanden. Doch wenn in der Projektarbeit alles gleich wie vorher ist, dann ist es keine Projektarbeit mehr, sondern Sachbearbeitung.

Die eingeleiteten Veränderungen, die KUBA beim Softwareproduzenten ausgelöst hat, sind in Gange und Erfolge liegen bereits vor. Das Beibehalten der Veränderung und das Verhindern des Rückfalls in alte Gewohnheiten auf allen Gebieten der Veränderung ist die Schwelle zu einem anhaltenden Erfolg [10].

4. Schlussfolgerungen

Was sind unsere Erkenntnisse aus dem dargestellten Projekt? Im Sinne der ‚lesson learned‘ sind folgende Punkte beachtenswert:

- Alle Projekte haben eine (Vor-)Geschichte, die berücksichtigt werden muss. Meist basieren die Softwareprojekte auf Optimierungspotenzialen, die mittels Analyse der Geschäftsprozesse erkannt werden. Die Instanz der Qualitätssicherung gewinnt an Bedeutung und bekommt vielfach die zusätzliche Rolle des Integrators bei der Anbindung an vorhandene Systeme. In diesem Umfeld nimmt die Qualität eine zentrale Rolle ein, denn die neuen vernetzten Systeme lassen eine direkte, schnelle und exakte Fehlerlokalisierung kaum mehr – oder noch nicht - zu.
- Die Software Qualität muss immer aus Sicht des Kunden betrachtet werden: Welche Qualitätsfaktoren sind für sein Business entscheidend? [15]
- Softwareentwicklungsprojekte sind umfassend und können nicht isoliert betrachtet werden. Ein sehr heterogenes Know-how ist gefordert, das nur durch das Zusammenspiel von mehreren Personen und Parteien eingebracht werden kann. In der Software Entwicklung ist der Mitarbeitende und sein Wissen erfolgskritisch [15]. Wenn jeder die neuesten Methoden und Technologien einsetzt, dann entscheidet die Qualität der Mitarbeitenden über den Geschäftserfolg.
- Die Anforderungen an die Teamarbeit nehmen zu. Die Kultur für Kritik, Offenheit und das Bekenntnis zu Fehlern müssen aktiv aufgebaut werden. Konflikte verdienen Respekt und sind kein Zeichen von unprofessionellem Verhalten. Die meisten Systementwicklungs-Organisation verfügen über schlechte Konfliktlösungsfähigkeiten [13]. Die Kritikfähigkeit als persönliche Herausforderung ist zu schulen und im Team zu fördern. Ein Coaching als weitere neutrale Stelle neben der Qualitätssicherung würde viel Abhilfe schaffen, so dass die Konflikte ausgetragen und bereinigt werden können, ohne einen Projektabbruch zu provozieren.
- Der Einsatz der Qualitätssicherung muss so früh und so umfassend wie möglich erfolgen. Sie soll nicht nur im abschliessenden Testing eingesetzt werden, sondern in allen Phasen eines Projektes. Um die Qualität in Projekten umfassend zu sichern, ist die Qualitätssicherung auf die Ebene der Projektleitung zu erheben und die gesamte Projektleitung bzw. das Projektmanagement mit den Methoden des Riskmanagements zu führen.

So eingesetzt wird die Qualitätssicherung zum äusserst erfolgreichen Mittel zur Erreichung der Ziele unter der Einhaltung der gesetzten Parameter Termin, Qualität, Funktionsumfang und Kosten.

5. Literaturhinweise

- [1] Wallmüller, Ernest, "Software-Qualitätssicherung in der Praxis", Carl Hanser Verlag, München, 1990, ISBN 3-446-15846-4
- [2] Frühauf K, Lideweg J., Sandmayr H., "Software-Projektmanagement und Qualitätssicherung", vdf Hochschulverlag AG ETH Zürich, Zürich, 1999
- [3] Malik, M.o.M., "Darf man Fehler machen?", Letter on Management 6/95, S.86-93, 1995
- [4] Prof. Dr. Schönsleben, Paul und Dr. Specker, Adrian, "Von der Softwareentwicklung zum Softwareunternehmen", iomanagement Nr. 6, 1999, S.90-96
- [5] Paulk, Mark Cc, "The capability maturity model guidelines for improving the software process", et al., 1995
- [6] Myers, G.J., "Methodisches Testen von Programmen", Oldenbourg Verlag München Wien, 1989, ISBN 3-486-21317-2

- [7] Software Engineering Standards, ANSI/IEEE-Standards 729, 730, 828, 829, 830, ISBN 471-82802-5
- [8] Moore, James W., "Software Engineering Standards, A User's Road Map", IEEE Computer Society, 1998, ISBN 0-8186-8008-3
- [9] Möller, K. H., Paulish D. J., "Software Metrics", Chapman & Hall, 1993, ISBN 0-412-45900-0
- [10] Häfelfinger, Kurt, "Grundsätze für ein erfolgreiches Change Management", Management & Qualität 6/99, 1999, S.58-61
- [11] Senge, Peter, "The Dance of Change", Doubleday, New York, 1999
- [12] de Marco, Tom and Lister Timothy, "Wien wartet auf dich – Der Faktor Mensch im DV-Management", Carl Hauser Verlag München Wien, 1991, ISBN 3-446-16229-1
- [13] de Marco, Tom, "The Deadline", Dorset House Publishing Co., Inc, New York, 1997, ISBN 3-446-19432-0
- [14] Fopp and Schiessl, "Business Change als neue Management-Disziplin", Campus, 1999, ISBN 3-593-36302-X
- [15] Prof. Dr. Yoshinori, Iizuka, "2nd World Congress for Software Quality", Japan, 2000