

Kommunikation und Softwarequalität

Vortrag im Rahmen des Seminars 'Qualität der Informatik' zum ADV-Kongreß Wien am 30.5.96

Dipl.-Mathematiker Wolfgang Ulbrich

SAG Systemhaus GmbH, Anwendungsentwicklung, Niederlassung München
Elsenheimerstr. 11, D-80687 München
Tel.089/ 5 47 42 - 188, Fax 089/ 5 47 42 - 199

These:

Die Erstellung qualitativ hochwertiger Software lebt wesentlich von der Beherrschung der dem Softwareentwicklungsprozeß immanenten Kommunikationsprobleme.

Im folgenden werden für ausgewählte Schwerpunkte der Softwareentwicklung einige dieser Kommunikationsprobleme benannt und pragmatische Lösungs-/Vermeidungs-Strategien diskutiert sowie auf der Basis praktischer Erfahrungen bewertet. Bei der Installation und erfolgreichen Anwendung von Qualitätsmanagementsystemen u.a. nach ISO 9000 ff. stellt die Kommunikation einen nicht zu vernachlässigenden Erfolgsfaktor dar.

1. Angemessene (semantisch und syntaktisch korrekte und vollständige sowie verständliche) Modellierung des zu lösenden 'Reale-Welt'-Problems

Bei der Softwareentwicklung wird in der Regel ein definierter Ausgangspunkt gesetzt, auf den sich letztendlich alle Bewertungen des Prozeßergebnisses beziehen. Diese Bewertungen beziehen sich z.B. sowohl informell auf die Erfüllung der Nutzererwartungen, als auch auf die Erfüllung vertraglich vereinbarter Entwicklungsziele mit möglichen Folgen für Bezahlung, Regreß- oder Gewährleistungsansprüche. Als solcher definierter Ausgangspunkt ist die 'reale Welt' selbst untauglich, da sie selbst bzw. mindestens ihre Wahrnehmung und Interpretation in ständiger Veränderung begriffen ist. Ein oft gewählter Weg ist deshalb die Modellierung des zu lösenden 'Reale-Welt'-Problems und die Festschreibung des Modells als Ausgangspunkt für die weitere Softwareentwicklung. Dies bedeutet nicht, daß das Modell während des Entwicklungsprozesses nicht mehr veränderbar wäre, aber solche Veränderungen müssen ebenfalls definiert erfolgen und vor allen Dingen festgehalten werden. Mögliche Ergebnisse der Modellierung sind im allgemeinen eine Spezifikation und/oder ein Prototyp der Lösung. Eine Bewertung dieser Möglichkeiten wird im Rahmen des Vortrags nicht vorgenommen. Ebenfalls wird im Vortrag nicht auf Probleme möglicher menschlicher Erkenntnisgrenzen hinsichtlich sehr komplexer Sachverhalte der 'realen Welt' eingegangen.

Folgende kommunikativen Probleme treten bei der Modellentwicklung u.a. auf :

- Entwickler und spätere Nutzer der Software sprechen eine unterschiedliche 'Sprache'.
- Gleiche Begriffe können in beiden Sprachräumen unterschiedliche Bedeutungen haben.
- Verwendete Begriffe können im jeweiligen Sprachraum mehrdeutig sein.
- Es existieren unterschiedliche Erwartungen an die Modelldarstellung bei Entwickler und Nutzer:
 - * Entwickler: Exaktheit (auch im Detail) und für die weiteren Entwicklungsprozesse geeignete Darstellung, d.h. evtl. Entity-Relationship-Modell o.ä.
 - * Nutzer: Verständlichkeit, Anschaulichkeit bei Verzicht auf 'unwesentliche(!?)' Details, d.h. seiner Begriffs- und Beschreibungswelt angenähert
- Man muß qualifizierte spätere Nutzer, d.h. Nutzer mit Kompetenz, Sachkunde und Bereitschaft zur gründlichen Analyse, finden, die das erarbeitete Modell beurteilen.

Damit nicht schon bei der Modellierung die Grundlage für ein unbefriedigendes Ergebnis des Softwareentwicklungsprozesses gelegt wird, muß man versuchen, die beschriebenen Probleme zu vermeiden bzw. ihnen geeignet zu begegnen.

Dafür gibt es u.a. folgende pragmatischen Möglichkeiten

- Die Entwickler eignen sich die wesentlichen fachlichen Elemente des Nutzersprachschatzes an und verwenden sie auch.
- Es erfolgt eine exakte Definition von Begriffen im Rahmen der Modellierung und deren Beschreibung in einem Projektlexikon.
- Die bei der Modellierung mitwirkenden Nutzer werden mit den verwendeten Modellierungstechniken vertraut gemacht.
- Es wird eine Ergänzung von Text durch nichttextuelle Elemente (Bilder, Grafik etc.) bzw. umgekehrt die Ergänzung verwendeter Diagrammtechniken durch textuelle Erläuterungen vorgenommen.
- Ein dynamisches Modell (Prototyp) zum simulierten Umgang mit den wesentlichen Elementen der späteren Lösung wird den Nutzern zur Verfügung gestellt.

2. Umsetzung (Verfeinerung) des Modells je nach vorgeschriebener Vorgehensweise und verfügbarer Entwicklungsumgebung

Dies geschieht in der Praxis in Schritten, Stufen, Ebenen, Phasen je nach begrifflicher Benennung. Im folgenden wird rein pragmatisch der Begriff 'Phase' verwendet.

Zwischen jeweils zwei Phasen A und B tauchen analoge Probleme zum ersten Schritt der Modellierung auf:

- Die Entwickler können in den Phasen unterschiedlich sein. Die Firma X hat beispielsweise das 'Fachkonzept' erstellt und von der Firma Y soll jetzt ein 'DV-Konzept' erarbeitet werden. Damit ist eine unterschiedliche 'Sprache' in Phase A und B mit Auswirkungen analog zu 1 vorhanden.
- Die Beschreibung des Modells ist meistens nicht eindeutig, da infolge der unter Pkt. 1 genannten Probleme selten mit den Mitteln einer streng formellen Spezifikation gearbeitet wird. Die vollständige Beschreibung bestünde in der Regel nicht nur aus den dokumentierten Ergebnissen sondern auch aus dem bei den Modellentwicklern angesammelten Hintergrundwissen (informelles Teamgedächtnis). Es gibt immer wieder implizite Prämissen des Modells, die nirgends dokumentiert sind. Diese Probleme zeigen sich besonders gravierend bei der Vergabe von Softwareentwicklungsaufträgen in 'Billiglohn'-Länder.
- Es muß eine Bewertung des Ergebnisses von Phase B bezüglich der Vorgaben durch das Ergebnis von Phase A bzw. noch weiter zurückliegender Phasen durch spätere Nutzer bzw. die Entwickler in Phase B erfolgen.
- Es können sich Fehler aus vorherigen Phasen fortpflanzen.

Folgende Maßnahmen zur Lösung bzw. Vermeidung der Probleme können ergriffen werden:

- Der wesentliche fachliche Sprachschatz der Phasen wird abgestimmt.
- Das Projektlexikon wird um die exakte Definition von Begriffen im Rahmen der aktuellen Phase ergänzt. Wenn bisher kein Projektlexikon existierte, so wird es angelegt. Unbedingt vermieden werden sollte eine Änderung von schon definierten Begriffen (Konsistenzprobleme!).
- Text (auch Programmcode) sollte sofern möglich durch nichttextuelle Elemente ergänzt werden.
- Sofern noch nicht vorhanden wird ein dynamisches Modell (Prototyp) zum simulierten Umgang mit den wesentlichen Elementen der späteren Lösung den Nutzern zur Verfügung gestellt bzw. weiterentwickelt und verfeinert.
- Die Entwicklung sollte so wenig Phasen wie möglich durchlaufen.
- Mindestens ein Entwickler/begleitender Nutzer sollte durch alle Phasen beteiligt sein, um das 'informelle Teamgedächtnis' nutzen zu können.
- Eine Prüfung von Phasenergebnissen sollte durch mehrere auch unabhängige 'Experten' in Reviews u.ä. erfolgen.

3. Softwareentwicklung im Team innerhalb einer Phase

Auch in einem gut organisierten und optimal zusammengesetzten Team treten Kommunikationsprobleme auf. Der entscheidende Unterschied zwischen erfolgreichen und weniger erfolgreichen Teams besteht u.a. darin, ob man diese Probleme erkennt und wie man mit ihnen umgeht.

Die häufigsten Kommunikationsprobleme sind:

- Die Entwickler sprechen eine unterschiedliche 'Sprache'.
- Es gibt im Team keine einheitliche Basis für die Entwicklung (Werkzeuge, Standards u.ä.).
- Es findet kein ausreichender Austausch über Probleme und/oder Ergebnisse im Team (Verbindung im Team) statt.
- Im Team existiert keine ausreichend saubere Aufgabenteilung (Abgrenzung im Team).

Was kann man empfehlen, um die genannten Probleme zu vermeiden bzw. zu lösen ?

- Auch im Team selbst sollte eine Abstimmung des wesentlichen Sprachschatzes stattfinden.
- Es ist notwendig Standards und Umgebungsbedingungen exakt zu definieren (soviel wie nötig, nicht soviel wie möglich!) und deren Einhaltung kontrollfähig zu machen. Dabei ist der Selbstkontrolle unbedingt der Vorrang vor der Fremdkontrolle zugeben. Wenn es ein Werkzeug zur statischen Analyse in der Entwicklungsumgebung gibt, so muß dies allen Entwicklern zur Verfügung stehen. Es muß auch bekannt sein, welche Kontrollen vom Projektmanagement für notwendig erachtet und deshalb durchgeführt werden.
- Ein regelmäßiger Austausch über Probleme und/oder Ergebnisse im Team (offiziell und informell) ist das Lebenselixier erfolgreicher Teamarbeit.
- Ein klares und systematisches Projektmanagement schafft Berechenbarkeit.
- Die gegenseitige Prüfung von Ergebnissen (Reviews u. ä.) ist in einem erfolgreichen Team selbstverständlich. Schuldzuweisungen sind ungeeignete Mittel. In der Sache eventuell harte aber im Ton moderate, auf keinen Fall persönlich verletzende und diffamierende Auseinandersetzung nützt mehr als das 'unter den Teppich kehren' von Problemen.
- Wenn irgend möglich, sollten sich die Teammitglieder über die Prinzipien erfolgreicher Teamarbeit und Kommunikation im Team informieren können, um die in einem Team bzw. generell zwischen Menschen wirkenden Kommunikationsmechanismen verstehen zu lernen.

4. Erkenntnis-/Kommunikations-Probleme beim einzelnen Entwickler

Letztendlich sind aber bei aller Teamarbeit durch den einzelnen Entwickler einzelne Aufgaben zu lösen. Dabei stößt er auch wieder auf Kommunikationsprobleme, die die Kommunikation mit sich selbst und die richtige oder vielleicht besser gesagt, der Aufgabe angemessene Interpretation der Einordnung seiner Lösung in den Gesamtzusammenhang beinhalten. Im Detail treten dabei auf:

- Der Entwickler macht Vorgehensfehler und gerät in 'Denkfallen' auf Grund seiner evolutionär bedingten und/oder durch Bildung angelernten Lösungsstrategien bzw. Lösungstechniken.
- Der Entwickler betreibt keine oder nur eine ungenügende Analyse der eigenen Fehler.
- Er lernt nicht oder nicht genügend oder nicht richtig aus seinen Fehlern.
- Er eignet sich nur ungern und mangelhaft 'erfolgreiche Techniken/best practices' an.'

Es ist schwierig, zu der Lösung dieser Probleme bei Achtung der Individualität und Persönlichkeit des einzelnen Entwicklers etwas zu tun, aber es ist nicht unmöglich, sondern es ist notwendig und auch erfolgversprechend

Man kann unter anderem

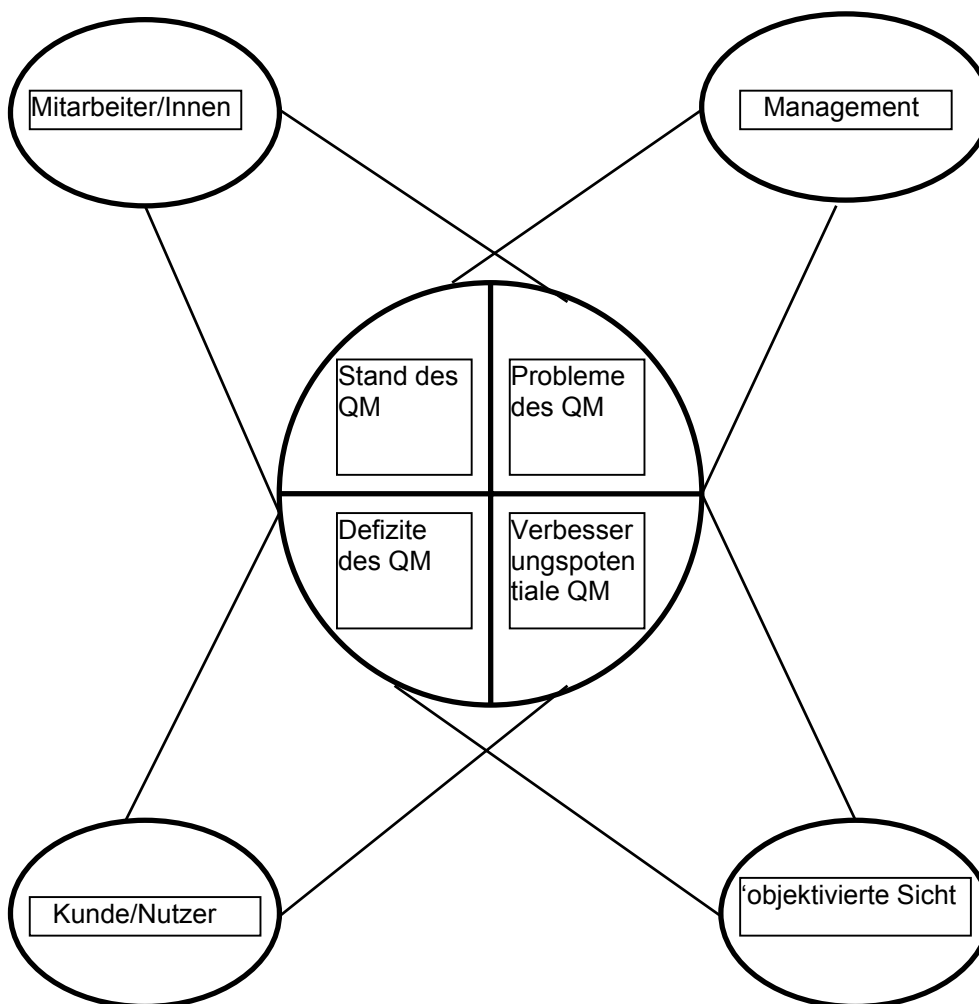
- die Entwickler über die Rolle psychischer Prozesse und Erkenntnisprozesse bei der Softwareentwicklung aufklären,
- die Anwendung von Analysetechniken und Metriken durch den Entwickler fördern und in bestimmtem Maße verlangen,

- moralisch und/oder materiell die Weitergabe und Aneignung 'erfolgreicher Techniken/best practices' im Team stimulieren

5. Unterschiedliche Sichtweisen auf Softwarequalität und Softwareentwicklung

Jeder erwartet zumindest von der von ihm selbst benutzten Software oder von der Software, für die man irgendwie verantwortlich ist, 'gute Qualität'. Dabei ist das Qualitätsverständnis der Beteiligten oft so unterschiedlich wie es unterschiedlicher nicht sein könnte. Dies gilt nicht nur für das Produkt (die fertige Software), sondern auch für den Prozeß, der dieses hervorbringt und ebenfalls für das Qualitätsmanagement des Softwareentwicklungsprozesses.

Qualitätsmanagement im Spiegel der Meinungen



Für ein erfolgreiches Projekt bzw. für eine erfolgreiche Softwareentwicklung im Unternehmen überhaupt ist die Kommunikation über die unterschiedlichen Meinungen sowie die Erzielung eines möglichst umfassenden Konsens über die Schwerpunkte ein wesentlicher Faktor der Verbesserung. Dazu muß man die unterschiedlichen Sichtweisen erst einmal kennen.

Welche Möglichkeiten gibt es hierfür ?

- Ein **Self-Assessment der Mitarbeiter** unter strikter Wahrung der Anonymität zeigt die subjektive Wahrnehmung der Qualitätssituation im Projekt/Unternehmen durch die Mitarbeiter.
- Ein **Self-Assessment des Managements** ebenfalls unter strikter Wahrung der Anonymität zeigt die subjektive Wahrnehmung der Qualitätssituation im Projekt/Unternehmen durch das Management.
- Eine **objektivierte Analyse**, z.B. durch Assessments nach Bootstrap oder Nutzung von Controlling-Methoden oder Audits zeigt den Stand bezüglich formulierter Ansprüche z.B. Reifestufen der Softwareentwicklung o.ä. und ermöglicht damit die Einordnung des eigenen Prozesses gegenüber den Mitbewerbern am Markt.
- Die **Befragung der Kunden/Nutzer** zeigt die Schwachpunkte des eigenen Prozesses, z.B. im Änderungsmanagement, Support und Verhalten gegenüber dem Kunden/Nutzer allgemein, aus Kunden-/Nutzersicht. Seltener werden positive Eigenschaften hervorgehoben.

Eine in intensiver Kommunikation auf der Basis dieser vier Grundmeinungen und einem gemeinsam erarbeiteten Konsens formulierte Verbesserungsstrategie muß wiederum intensiv an die o.g. Gruppen übermittelt werden. Sie muß vor allem die Einleitung akzeptierbarer, überschaubarer und pragmatischer Maßnahmen zum Inhalt haben, damit sie gelebt und zum Erfolg geführt werden kann. Gleichzeitig muß die Kontrolle der Wirksamkeit der Veränderungen gesichert werden, um keine Demotivierung der Beteiligten nach dem Motto 'es ändert sich ja doch nichts' zuzulassen.

6. Schlußbemerkung

In vielen Standardwerken zur Softwarequalität finden sich einzelne Aspekte der im Vortrag behandelten Kommunikationsprobleme. Leider beschränken sich die meisten Autoren auf eine mehr technische Sicht der Probleme. Software wird aber (noch!) von Menschen gemacht. Deshalb ist die Berücksichtigung der Möglichkeiten und Grenzen des Menschen nicht zuletzt in kommunikativer Hinsicht m.E. nach eine der wesentlichen Möglichkeiten zur Verbesserung der Softwarequalität. In diesem Vortrag konnten auf Grund des zur Verfügung stehenden Zeitlimits viele Dinge nur angerissen werden, die verdienten, wesentlich ausführlicher besprochen zu werden, wofür durch den Vortrag vielleicht eine Anregung gegeben wurde.

Literaturhinweise

Die Standardwerke zur Softwarequalität sollen hier nicht genannt werden. Jedes von ihnen enthält eine reiche Palette von weiteren Literaturhinweisen. Hier soll nur auf zwei Quellen speziell verwiesen werden, die die Problematik 'Kommunikation und Softwarequalität' etwas ausführlicher behandeln.

F.C.Brodbeck/M.Frese (Hrsg.): Produktivität und Qualität in Softwareprojekten. Psychologische Analyse und Optimierung von Arbeitsprozessen in der Softwareentwicklung. Oldenbourg Verlag München Wien 1994

Timm Grams. Denkfallen und Programmierfehler. Springer-Verlag Berlin Heidelberg 1990

W. Ulbrich beschäftigte sich bereits in seiner Diplomarbeit mit der Kompliziertheit von Algorithmen. Während seiner langjährigen Tätigkeit als Projektleiter in mehreren Firmen u.a. seit 1990 bei der Software AG sammelte er umfangreiche Erfahrungen bei der erfolgreichen Gestaltung von Softwareentwicklungsprozessen. In dieser Zeit galt sein Augenmerk stets besonders der Softwarequalität. Gegenwärtig ist er in der SAG Systemhaus GmbH, der deutschen Landesgesellschaft der Software AG, in deren Niederlassung München als Qualitätsmanager in der Anwendungsentwicklung tätig. In dieser Stellung berät er eigene Entwicklungsteams und Kunden in allen Belangen des Qualitätsmanagements und des IV-Controlling.