Interner Bericht

Kontinuierliche Software-Qualitätsverbesserung in der industriellen Praxis

Christiane Differding und Dieter Rombach

282 / 96

Fachbereich Informatik

Universität Kaiserslautern • Postfach 3049 • D-67653 Kaiserslautern

Kontinuierliche Software-Qualitätsverbesserung in der industriellen Praxis[1]

Christiane Differding, Dieter Rombach
AG Software Engineering
Fachbereich Informatik
Universität Kaiserslautern
{differdi, rombach}@informatik.uni-kl.de

Interner Bericht 282 / 96

Abstract

In der industriellen Praxis werden immer häufiger Verbesserungs- und Meßansätze zur Steigerung der Qualität von Software-Produkten und -Projektdurchführungen diskutiert. Dieser Artikel gibt eine Übersicht über potentielle Ansätze zur kontinuierliche Software-Qualitätsverbesserung: QIP, CMM und AMI. Aus dem Vergleich der Verbesserungsansätze geht hervor, daß u.a. zielorientiertes Messen eine integrale Technologie zur Verbesserung ist. Deshalb wird in diesem Artikel ein Ansatz für zielorientiertes Messen, der GQM-Ansatz, detaillierter diskutiert. Insbesondere wird auf die Anwendung in der Praxis eingegangen, wobei die Erfahrungen aus realen Projekten in Form von Richtlinien vorgestellt werden.

Der Artikel will Praktikern einen Einstieg in die Software-Qualitätsverbesserung mittels Messen vermittlen.

Herausgeber: AG Software Engineering
Leiter: Prof. Dr. H. Dieter Rombach

Kaiserslautern, Juni 1996

---

[1] Dieser Artikel wird auch veröffentlicht in C. Ebert und R. Dumke (Hrsg.), Software-Metriken in der Praxis, Springer Verlag, 1996

Kontinuierliche Software-Qualitätsverbesserung in der industriellen Praxis

Christiane Differding und Dieter Rombach, Universität Kaiserslautern

1 Übersicht

Ohne qualitativ hochwertige Software sind die meisten Produkte und Dienstleistungen des täglichen Lebens heute kaum noch denkbar. Automobile mit geringerem Benzinverbrauch, hohen Standards für Sicherheit und Umweltschutz oder verbesserten Diagnosemöglichkeiten sind ohne Software nicht realisierbar. Vergleichbares gilt für Verkehrsleitsysteme, automatische Fertigungssysteme, Haushaltsgeräte und Telekommunikation. Allerdings erfüllt Software selten die von traditionellen Produkten oder Dienstleistungen selbstverständlich erwarteten Qualitätsanforderungen. Die Gesamtqualität eines Produkts oder einer Dienstleistung ist nur so gut wie die Qualität der schwächsten Komponente, deswegen gilt es, die Qualität der Software entscheidend zu verbessern.

Für Qualitätsverbesserungen der Software und ihres Entwicklungsprozesses gelten vier Voraussetzungen: Erstens muß bekannt sein, wie gut die Qualität oder Produktivität zum gegenwärtigen Zeitpunkt ist, zweitens müssen alternative Maßnahmen, die eine Verbesserung versprechen, bekannt sein. Drittens müssen diese Maßnahmen in den Entwicklungsprozeß integriert werden, und viertens muß der Grad der resultierenden Verbesserung gemessen werden können.

Unter den genannten Voraussetzungen können drei Grade von Verbesserungen der Qualität unterschieden werden:

- einmalige Verbesserungen in einem Projekt;
- wiederholbare Verbesserungen über eine Serie von Projekten hinweg;
- nachweisbare Verbesserungen, die vor Projektbeginn im Rahmen der Projektakquisition belegt werden können.

Verbesserungen müssen dann nachweisbar sein, wenn z.B. im Kfz-Bereich Serienprodukte in millionenfacher Auflage zertifiziert werden sollen.

Im folgenden wird als Beispiel für eine Verbesserung die Einführung systematischer Inspektionen beschrieben: Im Software-Entwicklungsprozeß werden zu viele Fehler zu spät erkannt. Die späte Erkennung führt zu Behebungskosten, die exponentiell höher sind als die

Kosten, die bei frühzeitiger Erkennung und Behebung der Fehler angefallen wären. Daher werden systematische Inspektionen mit dem Ziel eingeführt, möglichst viele Fehler unmittelbar nach ihrem Auftreten zu erkennen und zu beheben.

Durch die Einführung von systematischen Inspektionen kann in einem Projekt eine einmalige Verbesserung erzielt werden. Die Verbesserung besteht z.B. darin, daß die Fehlerbehebungskosten viel geringer sind als vorher. Eine solche einmalige Verbesserung stellt ein Erfolgserlebnis eines Projektes dar. Allerdings ist unklar, ob in einem Folgeprojekt durch die Einführung der gleichen Technik die gleiche Verbesserung erzielt werden kann.

Die Wiederholbarkeit von Verbesserungen von einem Projekt zum nächsten ist nicht ohne weiteres möglich, weil Software entwickelt und nicht produziert wird. Deswegen verändern sich gewisse Einflußfaktoren wie Erfahrung der Projektmitglieder oder Programmiersprache von Projekt zu Projekt. Um die in einem Projekt (in unserem Beispiel durch Inspektionen) erzielte Verbesserung in anderen Projekten wiederholbar zu machen, müssen die von Projekt zu Projekt variablen Einflußfaktoren bestimmt werden. Dies erfordert entweder Experimente, in denen die Einflußfaktoren bezogen auf Inspektionen kontrolliert betrachtet werden, oder empirische Untersuchungen, die Inspektionen in mehreren Projekten betrachten. Sind die Einflußfaktoren bestimmt, können begründete Annahmen über die Effekte von Inspektionen unter bestimmten Kontextbedingungen getroffen werden. Damit werden Verbesserungen in Projekten, die diesen Bedingungen genügen, wiederholbar. Zum Beispiel wird bei der Durchführung einer Serie von Projekten festgestellt, daß für Projekte mittlerer Größe mit erfahrenen Entwicklern, die Produkte in der Programmiersprache C entwickeln, Inspektionen den in Abb. 1 dargestellten Einfluß auf die Anzahl der in verschiedenen Phasen entdeckten Fehler haben. Die Anzahl der Fehler, die ihre Ursache in den Anforderungen haben und in

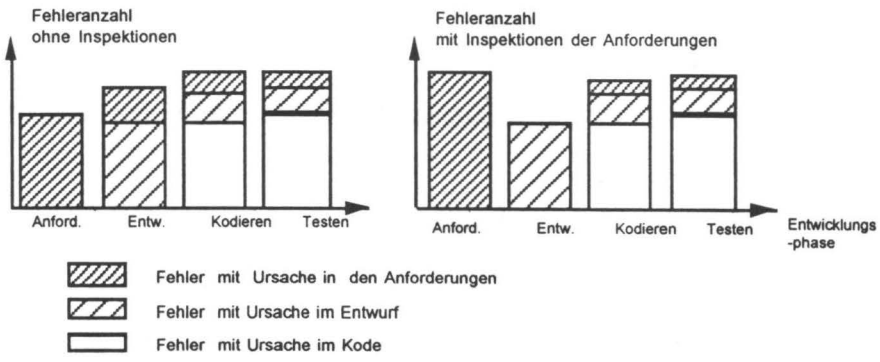


Abb. 1. Anzahl der Fehler nach Phasen ihrer Entstehung und Entdeckung mit und ohne Inspektionen

späten Phasen gefunden werden, sinkt. Dieses Wissen kann nun für weitere Projekte dieser Art verwendet werden. Die Verbesserung ist wiederholbar geworden.

Soll die eingeführte Verbesserung nicht nur wiederholbar, sondern auch von Dritten nachvollziehbar und somit nachweisbar sein, muß zusätzlich der im Projekt angewendete Entwicklungsprozeß explizit beschrieben werden. Falls die Verbesserung in der Durchführung von Inspektionen besteht, wird z.B. die genaue Vorgehensweise zum Lesen der Dokumente, die Beschreibung des Ablaufs der Sitzungen und die Form der erstellten Fehlerprotokolle festgehalten. Damit gibt es eine explizite Beschreibung der Vorgehensweise, die zu der Verbesserung geführt hat.

In diesem Kapitel wird ein Ansatz für kontinuierliche Qualitätsverbesserung vorgestellt, der zur Erreichung aller drei Grade von Verbesserung eingesetzt werden kann. Dieser Ansatz [Basi95] wird auch als „Experimenteller Software-Engineering-Ansatz" bezeichnet und wird bereits mit Erfolg praktiziert. Kap. 2 beschreibt die wissenschaftlichen Grundlagen kontinuierlicher Qualitätsverbesserung und stellt verschiedene wissenschaftliche Vorgehensweisen einander gegenüber. Kap. 3 erläutert die zur Verbesserung benötigten Technologien. Wir konzentrieren uns insbesondere auf das Messen. Kap. 4 beschreibt den Start eines Qualitätsverbesserungsprogramms nach dem Ansatz des Experimentellen Software-Engineering. Kap. 5 enthält Beispiele von Organisationen, die den Experimentellen Software-Engineering-Ansatz anwenden. Kap. 6 faßt die Grundgedanken noch einmal zusammen.

2 Kontinuierliche Qualitätsverbesserung

Kontinuierliche Software-Qualitätsverbesserung erfordert systematische Maßnahmen zur Formulierung und zur Wiederverwendung von explizitem Software-Entwicklungs-Know-how. Dieses Software-Entwicklungs-Know-how variiert von Unternehmen zu Unternehmen entsprechend deren spezifischer Ziele und Umgebungscharakteristika. Es ist einsichtig, daß das Know-how zur Erstellung von Software für Banken in Cobol sich von der Erstellung von Software zur Prozeßsteuerung in C unterscheidet.

Aufbau und Verbesserung von Know-how wird in verschiedenen Disziplinen betrieben. Deshalb werden in Kap. 2.1 existierende Vorgehensweisen unterschiedlicher Disziplinen beschrieben. Kap. 2.2 gibt Beispiele für explizites Software-Entwicklungs-Know-how. Kap. 2.3 gibt eine Übersicht über den von uns verfolgten Ansatz zur kontinuierlichen Qualitätsverbesserung, das Quality Improvement Paradigm. Kap. 2.4 beschreibt weitere Ansätze zur Qualitätsverbesserung. Kap. 2.5 enthält einen Vergleich der verschiedenen Ansätze.

2.1 Prinzipien und Grundlagen

In wissenschaftlichen Disziplinen werden abhängig vom Untersuchungsgegenstand unterschiedliche Vorgehensweisen zum Aufbau von Wissen gewählt. Die Physik benötigt andere Vorgehensweisen als beispielsweise die Medizin oder das Bauingenieurwesen. Um zu ent-

schieden, welche Vorgehensweisen sich zum Aufbau von Software-Entwicklungs-Know-how eignen, müssen zunächst die Eigenschaften von Software betrachtet werden [Basi95]:

- Software ist ein aus unsichtbaren Teilen zusammengesetztes logisches Aggregat. Die Qualität eines solchen Aggregates hängt von der Güte und der Dokumentation der logischen Strukturierung der einzelnen Teile ab.
- Die Anforderungen an Software entwickeln sich kontinuierlich weiter. Eine wichtige Qualitätseigenschaft von Software ist daher die Anpaßbarkeit existierender Teile an ein sich änderndes Umfeld.
- Software wird entwickelt und nicht produziert. Jedes Produkt ist ein Unikat. Qualitätsanalysen bezogen auf ein Produkt sind daher nicht ohne weiteres auf andere Produkte übertragbar.
- Software-Entwicklung ist eine Technologie, die stark von den beteiligten Menschen abhängt. Die Qualität des entwickelten Produktes ist abhängig von den Kenntnissen, der Zufriedenheit und der Motivation der Entwickler.
- Es besteht ein Mangel an Modellen zur Unterstützung der Analyse von Software-Produkten und ihren Entwicklungsprozessen. Dies verringert die Vorhersagbarkeit der Eigenschaften von Software und ihren Entwicklungsprozessen.

Die geschilderten Eigenschaften von Software und ihrer Entwicklung führen dazu, daß existierende Ansätze zur Qualitätsverbesserung von Produktionsgütern nur bedingt für Software gelten. Total Quality Management (TQM) kann für Software nicht ohne weiteres angewendet werden, weil es auf dem mehrmaligen Durchführen von Produktionsprozessen und dem daraus ableitbaren Feedback basiert. So können Verbesserungen bei jeder erneuten Durchführung des Prozesses eingebracht werden. Die Einmaligkeit von Software-Projekten erfordert es hingegen, die Unterschiede zwischen Projekten zu beachten. Es muß verstanden werden, durch welche Charakteristika Projekte sich unterscheiden, damit Know-how über Projektgrenzen hinweg genutzt werden kann. Der Aufbau von Know-how muß zusätzlich zur Durchführung von Projekten erfolgen, damit später durchgeführte Projekte auf dieses Know-how zugreifen können.

Welche Vorgehensweisen können zum Aufbau von Software-Entwicklungs-Know-how angewendet werden? Wir betrachten Vorgehensweisen zum Aufbau von Know-how aus anderen Disziplinen. Ingenieurdisziplinen wie das Bauingenieurwesen wenden die ingenieurwissenschaftliche Vorgehensweise an, Disziplinen wie die experimentelle Physik verwenden die empirische Vorgehensweise, in der Mathematik wird die mathematische Vorgehensweise angewendet. Im folgenden wird jede dieser Vorgehensweisen beschrieben, außerdem wird ihre Anwendung im Bereich des Software-Engineering angegeben [Basi92].

1. **Die ingenieurwissenschaftliche Vorgehensweise**: Betrachte die bereits existierenden Lösungen, schlage bessere Lösungen vor, realisiere sie, messe und analysiere.

Wiederhole diesen Prozeß, bis keine weiteren Verbesserungen möglich erscheinen. Diese Vorgehensweise ist ein evolutionär orientierter Verbesserungsansatz, bei dem davon ausgegangen wird, daß bereits Modelle des Software-Prozesses, der Produkte, der beteiligten Personen und der Umgebung existieren. Diese Modelle werden verändert, um sie zu verbessern. Unverzichtbarer Bestandteil dieser Methode ist die Notwendigkeit von Analyse und Messungen.

2. **Die empirische Vorgehensweise**: Schlage ein Modell vor, entwickle statistische und qualitative Methoden und wende diese Methoden in Fallstudien an. Messe und analysiere, validiere das Modell und wiederhole dieses Vorgehen. Diese empirische Vorgehensweise ist ein revolutionär orientierter Verbesserungsansatz. Bei diesem Ansatz wird damit begonnen, ein neues Modell oder eine neue Methode vorzuschlagen. Das neue Modell oder die Methode muß nicht auf bereits existierenden Modellen aufbauen. Das Modell oder die Methode wird nun auf die Effekte hin untersucht, die auf ein Produkt oder einen Prozeß ausgeübt werden. Auch bei dieser Vorgehensweise ist Messen und Analysieren unverzichtbar. Es reicht nicht, eine neue Methode oder ein neues Modell vorzuschlagen, es muß der Beweis dafür geliefert werden, daß diese Methode oder dieses Modell einen Vorteil gegenüber den bisher beschriebenen hat.
3. **Die mathematische Vorgehensweise**: Schlage eine formale Theorie oder eine Menge von Axiomen vor, entwickle eine Theorie, leite Resultate ab und, wenn möglich, vergleiche sie mit empirischen Beobachtungen. Diese Vorgehensweise ist deduktiv analytisch. Sie stellt einen Rahmen bereit, innerhalb dessen Modelle von Prozessen und Produkten entwickelt werden können und deren Schranken durch Modellmanipulationen festgestellt werden können.

Beim Aufbau und der Wiederverwendung von Software-Entwicklungs-Know-how sind alle drei Vorgehensweisen notwendig. Mittels der ingenieurwissenschaftlichen Vorgehensweise werden aufbauend auf den bereits existierenden Erfahrungsschatz bereits bestehende Techniken variiert oder an anderen Stellen eingesetzt und so weiterentwickelt. Um gesicherte Aussagen über die Effekte einer für eine Umgebung völlig neuen Technik treffen zu können, muß die empirische Vorgehensweise verwendet werden. Die Effekte der Anwendung der neuen Technik in einem Projekt sind abhängig von den Faktoren, die das Projekt charakterisieren. Experimente werden durchgeführt und währenddessen werden diese für die Technik relevanten Einflußfaktoren kontrolliert. Durch diese Experimente werden die Abhängigkeiten zwischen den Einflußfaktoren und den Effekten der Technik festgestellt. Die aus Projekten und aus Experimenten gewonnenen Erkenntnisse werden in Modelle gefaßt und abgelegt. Diese Modelle können mit Hilfe der mathematischen Vorgehensweise analysiert werden. So werden Vorgehensweisen verschiedener wissenschaftlicher Disziplinen kombiniert, um Software-Entwicklungs-Know-how aufzubauen und zu verbessern.

2.2 Explizites Software-Entwicklungs-Know-how

Explizites Software-Entwicklungs-Know-how besteht aus verschiedenen Arten von Modellen: Qualitätsmodellen und Baselines, Prozeßdefinitionen, Produkten und Erfahrungspaketen [Nati94]. Mit Hilfe solcher Modelle wird das Verständnis für ablaufende Software-Entwicklungsprozesse verbessert, die Durchführung von Projekten wird unterstützt und der Nachweis von Verbesserungen ermöglicht.
Qualitätsmodelle und Baselines enthalten quantitative Informationen, wie z.B. Kosten- und Ressourcenmodelle, Fehlerraten, Fehlertypen und Fehlervorhersagen, Produktgrößen, Änderungsarten, Zuverlässigkeitsmodelle und Änderbarkeitsmodelle. Abb.2 zeigt Beispiele für Qualitätsmodelle.

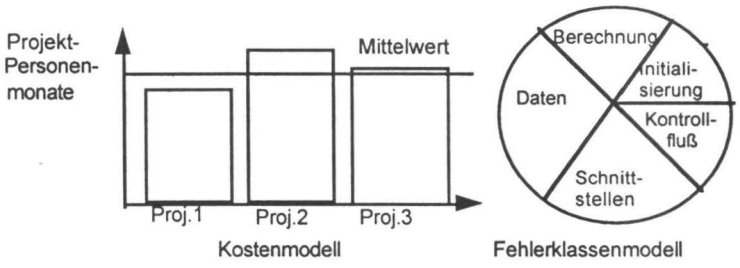


Abb. 2. Qualitätsmodelle für Kosten und Fehlerklassen

Prozeßdefinitionen beinhalten Prozeßmodelle für unterschiedliche Entwicklungsbereiche und Projektarten. Abb. 3 zeigt ein Beispiel für ein Prozeßmodell. Die abgelegten Produkte bestehen aus Rahmendokumenten für unterschiedliche Anwendungen, Trainingsmaterialien, Handbüchern etc. Erfahrungspakete beinhalten gesammelte Erfahrungen über bestimmte Themen. Ein Beispiel für eine Erfahrung mit Testtechniken ist: Lesen mit schrittweiser Abstraktion ist geeigneter für das Finden von Fehlern als strukturelles Testen.

Abb. 3. Prozeßmodell für eine Entwicklung nach dem Wasserfall-Modell

Der Genauigkeitsgrad des bisher aufgebauten Know-hows bestimmt den für das abgelegte Know-how gewählten Formalitätsgrad. Er stimmt überein mit den Entwicklungstechniken, die in den Projekten angewendet werden.

2.3 Quality Improvement Paradigm

Das Quality Improvement Paradigm (QIP) ist ein Ansatz zur kontinuierlichen Verbesserung und zur systematischen Durchführung von Software-Projekten. Es bildet die Grundlage des

Experimentellen Software-Engineering-Ansatzes. Das QIP wurde von Victor Basili und Dieter Rombach in Zusammenarbeit mit dem Software Engineering Laboratory (SEL) der NASA entwickelt [Basi85, BaRo88]. Das QIP kombiniert die ingenieurwissenschaftliche und die empirische Vorgehensweise und paßt sie den Bedürfnissen des Software-Engineering an. QIP kann als die auf die speziellen Charakteristika von Software angepaßte Form des TQM betrachtet werden. Es besteht aus sechs Schritten, die zyklisch ablaufen, siehe Abb. 4. Beim Vorgehen nach QIP kann jedes Projekt als Experiment betrachtet werden, bei dem vorhandenes Know-how wiederverwendet und neues Know-how gewonnen werden kann.

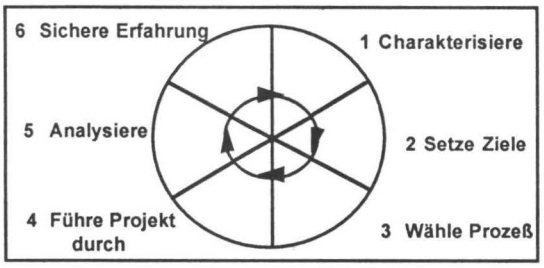


Abb. 4. QIP-Zyklus

Das Vorgehen nach dem QIP kann je nach Blickwinkel entweder auf die Durchführung eines Software-Projektes angewendet werden oder auf die Verbesserung des Know-hows der Organisation. Unter Organisation verstehen wir in diesem Zusammenhang die Gesamtheit der Software-entwickelnden Einheiten eines Unternehmens.

Tabelle 1 enthält die methodische Qualitätsverbesserungssicht des QIP. Die linke Spalte beschreibt die einzelnen Schritte aus der Sicht eines Projektes (Durchführung von Projekten), die rechte Spalte beschreibt sie aus der Sicht der Organisation (Verbesserung des Know-hows). Schritt 4 der Verbesserung des Know-hows umfaßt die Durchführung von in der Regel mehreren Projekten. Das QIP beschreibt das Vorgehen bei der Umsetzung von Verbesserungen in Software-Projekten. Die Einmaligkeit von Software-Projekten erfordert eine projektübergreifende Betrachtung von Verbesserungen. Die einzelnen Projekte können eine solche Betrachtung nicht leisten. Deswegen sind zusätzlich zu den Projekten organisatorische Strukturen notwendig, in denen der systematische Verbesserungsprozeß abläuft. Zur Unterstützung von Aufbau und Wiederverwendung von Know-how wird innerhalb der Software-Organisation eine sogenannte „Experience Factory" aufgebaut [BaRC94]. Diese „Experience Factory" ist logisch oder physisch von der Projektorganisation getrennt und unterstützt diese durch die Ermittlung und Bereitstellung von explizitem Know-how. Dieses Know-how ist in einer sogenannten Erfahrungsdatenbank abgelegt. Abb. 5 zeigt die Aufgaben der beiden Organisationsteile.

Tabelle 1. Methodische Qualitätsverbesserungssicht nach QIP

Durchführung von Projekten	Phase	Verbesserung des Know-hows
Charakterisiere Projekt und identifiziere existierendes Know-how	QIP 1 Charakterisiere	Erfasse Ist-Zustand und künftige Trends
Definiere Ziele des Projektes in meßbarer Form	QIP 2 Setze Ziele	Stelle Verbesserungsziele und Hypothesen auf
Wähle geeignete Prozesse und stelle Projektplan zusammen	QIP 3 Wähle Prozeß	Identifiziere geeignete Projekte oder Experimente zum Testen der Hypothesen
Führe Projekt gemäß Plan durch, erfasse und verwende Daten zur Projektkontrolle	QIP 4 Führe Projekt durch	Führe Pilotprojekte oder Experimente durch
Analysiere den Projektverlauf und schlage Verbesserungen vor	QIP 5 Analysiere	Werte Ergebnisse aus
Sichere Erfahrungen dauerhaft	QIP 6 Sichere Erfahrung	Bringe Ergebnisse als Know-how in künftige Projekte ein

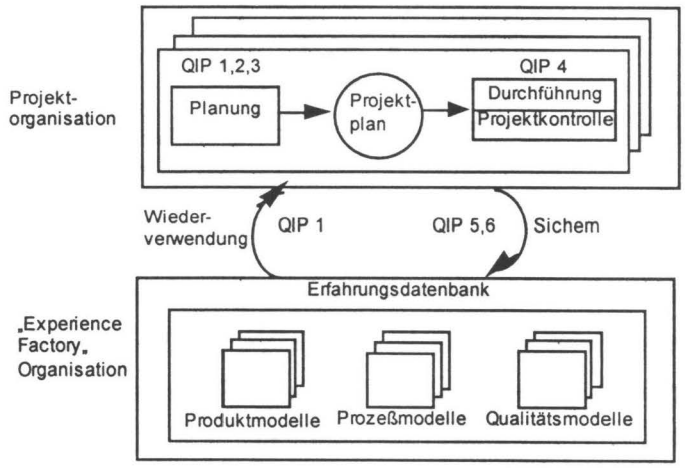


Abb. 5. Organisatorische Qualitätsverbesserungssicht nach QIP

Das QIP beinhaltet zwei Feedback-Zyklen, die Informationen unterschiedlicher Art und zu unterschiedlichen Zwecken liefern. Der projektspezifische Feedback-Zyklus liefert Informationen, die zur Projektkontrolle dienen. Sie unterstützen das Erreichen von Projektzielen und das rechtzeitige Erkennen auftretender Probleme. Dieser Feedback basiert auf dem Vergleich der Ist-Daten des Projektes mit den Soll-Daten aufgrund existierender Qualitätsmodelle. Der projektspezifische Feedback-Zyklus läuft während der Durchführung des Projektes ab, vergleiche QIP 4 in Tabelle 1.

Der organisatorische Feedback-Zyklus enthält den Feedback, der der gesamten Software-Organisation zur Verfügung steht, vergleiche QIP 5, 6 in Tabelle 1. Er erfüllt zwei verschiedene Aufgaben. Zunächst stellt er post-mortem Informationen über Projekte zur Verfügung, indem die Daten, die während der Durchführung von Projekten erfaßt wurden, verglichen werden mit den Modellen und Daten, die in der Erfahrungsdatenbank bereits abgelegt sind. Unterschiede und Gemeinsamkeiten werden festgestellt. Neben der Post-Mortem-Information über Projekte wird wiederverwendbare Erfahrung in Gestalt von Software-

Produkten und -Prozessen in der Erfahrungsdatenbank abgelegt. Die dort bereits existierenden Modelle werden validiert und verbessert.
Zur effektiven Umsetzung des QIP werden unterschiedliche Technologien benötigt. Das folgende Szenario entlang der Durchführung eines Projektes nach dem QIP beschreibt die Vorgehensweise und ihre Unterstützung durch die verschiedenen Technologien.

Schritt 1: Charakterisiere das Projekt und identifiziere existierendes Know-how. Das Projekt wird quantitativ charakterisiert, z.B. soll ein Produkt nach dem Cleanroom-Prozeßmodell in der Sprache C entwickelt werden. Fünfzig Personenjahre sind für das Projekt veranschlagt. Das Produkt ist ein eingebettetes Realzeitsystem von einer Größe von 100 KByte. Mit diesen Charakteristika werden in der Erfahrungsdatenbank Objekte identifiziert, die möglicherweise wiederverwendet werden können. Es wird nach Qualitätsmodellen gesucht, die ähnliche Produkte und Prozesse betreffen, nach der Prozeßdefinition für das Cleanroom-Prozeßmodell, und nach Erfahrungen mit der Sprache C bei der Entwicklung von Realzeitsystemen. Die in der Erfahrungsdatenbank gefundenen Objekte (Prozeßmodelle, Produktmodelle, Qualitätsmodelle und Erfahrungen) werden für das Projekt bereitgestellt.
Schritt 2: Definiere die Ziele des Projektes in meßbarer Form. Für das durchzuführende Projekt werden quantitative Ziele festgelegt. Der Projektaufwand soll fünfzig Personenjahre nicht überschreiten und im auszuliefernden Produkt sollen maximal fünf Fehler enthalten sein. Zur Überprüfung dieser Ziele während der Durchführung des Projektes werden Maße festgelegt. Beispiele für solche Maße sind die Anzahl und Art der Fehler, die bei Inspektionen und Tests gefunden werden.
Schritt 3: Wähle geeignete Prozesse und stelle Projektplan zusammen. In diesem Schritt wird ein Projektplan erstellt, der es erlaubt, die Projektziele zu erreichen. Der Projektplaner integriert die von der „Experience Factory" bereitgestellten Prozeßmodelle in den Projektplan. Gleichermaßen werden auch die Meßpunkte in den Projektplan integriert. Die Soll-Daten für Produkte und Prozesse werden entsprechend der bereitgestellten Qualitätsmodelle eingearbeitet. So ergibt sich z.B. aus der Zielvorgabe „maximal fünf Fehler" und dem entsprechenden Qualitätsmodell für Fehler analog zu Abb. 1 eine bestimmte Anzahl entdeckter Fehler auf die verschiedenen geplanten Inspektionen und Tests.
Schritt 4: Führe das Projekt gemäß Plan durch, erfasse und verwende Daten zur Projektkontrolle. Das Projekt läuft entsprechend dem Projektplan ab. Die Daten werden zu den definierten Zeitpunkten erfaßt und mit den Soll-Daten verglichen. Falls der Soll/Ist-Datenvergleich nicht zur Zufriedenstellung der Projektleiterin ausfällt, können frühzeitig Konsequenzen für das Projekt gezogen werden. Wenn zum Beispiel die Ist-Fehleranzahl in einem Test im Vergleich zur Soll-Fehleranzahl zu hoch ist, können Rückschlüsse auf die in diesem Projekt erreichbare Produktqualität gezogen werden. Noch während des Projektes können Konsequenzen gezogen werden.

Schritt 5: Analysiere post-mortem den Projektverlauf und schlage Verbesserungen vor. Der Projektablauf und das Projektergebnis werden analysiert und bewertet. Aus der Analyse hervorgehende Erfahrungen werden formuliert. Eine mögliche Erfahrung ist z.B.: Für die gegebenen Projektcharakteristika war die Inspektionstechnik X weniger effektiv als erwartet.

Schritt 6: Sichere Erfahrungen dauerhaft. Die in der Erfahrungsdatenbank abgelegten Modelle werden erweitert, z.B. kann ein bereits vorhandenes Qualitätsmodell validiert oder verändert werden, oder es kann ein neues hinzukommen. So kann z.B. die Fehlerverteilung für Projekte der gegebenen Art mit den in dem abgelaufenen Projekt neu gewonnenen Daten aktualisiert werden.

Eine Übersicht über die während der verschiedenen Schritte des QIP benötigten Technologien zielorientiertes Messen, explizites Modellieren und umfassende Wiederverwendung ist in Tabelle 2 enthalten. Diese drei Technologien werden im folgenden Infrastrukturtechnologien genannt. Kap. 3 geht genauer auf sie ein.

Tabelle 2. Einsatz der Infrastrukturtechnologien in Projekten

QIP	Zielorientiertes Messen	Explizites Modellieren	Umfassendes Wiederverwenden
1	Projekt quantitativ charakterisieren		Mit den Projektcharakteristika in Erfahrungsdatenbank suchen
2	Projektziele quantitativ definieren, Maße festlegen		
3	Zeitpunkt der Datenerhebungen fixieren, Solldaten vorgeben	Prozesse und Produkte modellieren, Modelle in Projektplan integrieren	
4	Ist-Daten erheben, Soll / Ist-Daten vergleichen	Projektplan ausführen, Ausführung kontrollieren	
5	Post-Mortem-Analyse und Interpretation durchführen	Projektdurchführung analysieren	
6			Erfahrungsdatenbank manipulieren

2.4 Weitere Ansätze zur Software-Qualitätsverbesserung

Neben dem in Kap. 2.3 vorgestellten QIP erheben eine Anzahl anderer Ansätze vergleichbare Ansprüche bezüglich der mit ihnen erreichbaren Qualitätsverbesserungen. In diesem Abschnitt werden typische Vertreter vorgestellt und bewertet.

2.4.1 Ad hoc

Der Ansatz, Verbesserungen ad hoc zu realisieren, ist der zur Zeit wohl am häufigsten verfolgte „Änderungs"-Ansatz. Änderungen der Software-Entwicklungsprozesse werden basierend auf der Intuition von erfahrenen Projektmanagerinnen oder Entwicklern eingeführt. Die eingeführten Veränderungen sind meist nur mangelhaft begründet, da keine systematischen Messungen über tatsächliche Schwächen vorab durchgeführt wurden. Häufig werden Methoden oder Techniken in Projekte eingeführt, die Vorteile versprechen, aber nicht unbedingt den in diesem Projekt vorhandenen Problemen angepaßt sind (z.B. Ein-

führung einer neuen Inspektionstechnik, obwohl nicht klar ist, ob sie zum Finden der am häufigsten auftretenden Art von Fehlern geeignet ist). Falls auf diese Weise eine neue Technik mit Erfolg eingeführt wird, ist nicht klar, ob der Erfolg durch die neue Technik selbst begründet war, oder ob andere projektabhängige Faktoren für den Erfolg verantwortlich waren, z.B. ein sehr gut zusammenarbeitendes Team oder eine besondere Art von Endprodukt. Deswegen kann über den Erfolg in weiteren Projekten wenig ausgesagt werden.

2.4.2 Capability Maturity Model (CMM)

Das Capability Maturity Model wurde von Likert und Crosby für allgemeine Prozesse entwickelt und von Radice und Mitarbeitern bei IBM an Software-Prozesse angepaßt. Populär wurde es durch Watts Humphrey beim Software Engineering Institute (SEI). Das CMM teilt den Reifegrad der Software-Entwicklung einer Organisation in fünf Ebenen ein. Die Einteilung basiert auf den Charakteristika der Entwicklungsprozesse, die in der Organisation ablaufen. Die von einer Organisation erreichte Ebene wird durch Assessments ermittelt. Es wird unterstellt, daß der Reifegrad der Organisation um so höher ist, je höher die bei einem Assessment erreichte CMM-Ebene ist. Die fünf Ebenen des CMM sind:

Ebene 5: Optimized: Kontinuierliche Prozeßverbesserung
Ebene 4: Managed: Kontrolle der Prozeßvariationen
Ebene 3: Defined: Installation eines allgemeinen Prozesses
Ebene 2: Repeatable: Stabilisierung der Umgebung
Ebene 1: Initial: Ad hoc Prozesse

Jede Ebene ist charakterisiert durch unterschiedliche „key process areas" (KPA), die sich auf die Software-Entwicklungsprozesse einer Organisation beziehen [PCCW93]. Die KPAs der verschiedenen Ebenen sehen folgendermaßen aus, wobei die Ebene n immer auch die KPAs der Ebene n-1 umfaßt.

Ebene 5:
- Fehlervermeidung
- Management von Technologieänderungen
- Management von Prozeßänderungen

Ebene 4:
- Quantitatives Prozeßmanagement
- Software-Qualitätsmanagement

Ebene 3:
- Fokussierung auf den allgemeinen Prozeß der Organisation
- Definition des allgemeinen Prozesses der Organisation
- Integriertes Software-Management
- Software-Produkt-Engineering
- Koordination zwischen Gruppen
- Trainingprogramme
- Peer Reviews

Ebene 2:
- Anforderungsmanagement
- Software-Projekt-„Tracking“
- Software-Subkontraktor-Management
- Software-Qualitätssicherung
- Software-Konfigurationsmanagement

Verbesserungsziel bei diesem Ansatz ist das „Hocharbeiten“ von der aktuell erreichten Ebene auf die nächsthöhere Ebene, bis die Ebene 5, die Ebene der kontinuierlichen Prozeßverbesserung, erreicht ist. Die jeweils aktuelle Ebene wird durch Assessments bestimmt. Zum Erreichen einer Ebene müssen die KPAs dieser Ebene unterstützt sein. Unterhalb der Ebene 5 erfolgen Verbesserungen bei diesem Ansatz, indem die für die nächsthöhere Ebene relevanten KPAs eingeführt werden.

2.4.3 Application of Metrics in Industry (AMI)

Der AMI-Ansatz wurde in einem ESPRIT-Projekt der Europäischen Union entwickelt. Er basiert auf einem zwölfstufigen Prozeß, dessen Stufen im folgenden angegeben sind. Sie sind eingeteilt in die verschiedenen Schritte Assess, Analyze, Metricate und Improve, [AMI92].

Assess:
- Bewerte die Umgebung
- Definiere Zielstellungen
- Validiere Zielstellungen

Analyze:
- Verfeinere Zielstellungen in Subziele
- Überprüfe die Konsistenz des resultierenden Zielebaums
- Produziere eine Tabelle mit Fragen zur Identifizierung von Maßen

Metricate:
- Schreibe und validiere den Meßplan
- Sammle Daten
- Verifiziere die Rohdaten

Improve:
- Verteile, analysiere und bespreche die Meßdaten
- Validiere die Maße
- Setze die Daten in Bezug zu Zielen und verwirkliche Maßnahmen.

Der AMI-Ansatz schlägt als Bewertungsmethode das CMM vor und als Meßansatz eine Abwandlung des GQM-Ansatzes [BaCR94]. Wichtige Prinzipien des GQM-Ansatzes (vergleiche Abschnitt 3.1.1) wurden bei der Anpassung an den AMI-Ansatz verletzt. Der AMI-Ansatz konzentriert sich auf die Bewertung des Standes der Software-Entwicklung und die Einführung von Messungen.

2.5 Vergleich der Ansätze

In diesem Abschnitt werden die beschriebenen Verbesserungsansätze auf ihre Eignung zum systematischen Aufbau von Know-how gegenüberstellt. Hierzu wird die Anwendung der in Kap. 2.1 eingeführten wissenschaftlichen Vorgehensweisen innerhalb des Ansatzes bewertet. Außerdem wird die Verwendung der verschiedenen Infrastrukturtechnologien Messen, Modellieren und Wiederverwendung innerhalb des Ansatzes untersucht. Zusammenfassend wird bewertet, inwieweit sich mit den Ansätzen die in Kap. 1 eingeführten Grade von Verbesserungen erreichen lassen.

Beim Ad-hoc-Ansatz werden Verbesserungen unsystematisch eingeführt. Der ingenieurwissenschaftliche Ansatz wird in dem Sinne angewendet, daß bezogen auf (ein meist implizites) Modell Veränderungen eingeführt werden, mit dem Ziel, Prozeß oder Produkt zu verbessern. Meist unterbleibt die Erfolgskontrolle. Nicht zur Anwendung kommt hier der empirische Ansatz, mit dem neue Vorgehensweisen vor der Einführung in Projekte bewertet werden. Messen ist die Grundlage für die Identifikation von Verbesserungen und wird daher vorausgesetzt. Die Nutzung von expliziter Modellierung ist nicht vom Ansatz vorgegeben, genausowenig wie die Wiederverwendung. Der Ad-hoc-Ansatz kann zu beachtlichen projektspezifischen Verbesserungen führen. Allerdings werden diese Verbesserungen im allgemeinen nicht wiederholbar sein, weil die Einflußgrößen, die bei der Verbesserung eine Rolle gespielt haben, nicht bekannt sind. Auch die Nachweisbarkeit der Verbesserungen a priori ist nicht möglich.

Im CMM wird die ingenieurmäßige Vorgehensweise als Grundlage für die Einführung von Verbesserungen unterhalb der Ebene 5 nicht angewendet. Die empirische Vorgehensweise wird gar nicht angewendet. Die Entwicklung der Organisation von einer niedrigeren Ebene bis zur Ebene 5 läuft nicht nach ingenieurwissenschaftlicher oder empirischer Vorgehensweise ab, sondern nach dem spezifischen Ebenenmodell des CMM. Da Software-Prozesse jedoch von Umgebung zu Umgebung stark variieren, muß dieses Modell nicht für jede Software-Organisation sinnvoll sein. Auf der Ebene 5 wird der ingenieurwissenschaftliche Ansatz zur ständigen Optimierung angewendet. Die empirische Vorgehensweise ist nicht Bestandteil des CMM-Ansatzes. Unterhalb von Ebene 4 ist Messen kein expliziter Bestandteil der Verbesserungsaktivitätenund wird erst auf Ebene 4 als KPA explizit gefordert. Explizit modelliert werden im CMM-Ansatz ab Ebene 3 nur Prozesse. Umfassende Wiederverwendung wird innerhalb des CMM-Ansatzes nur in Bezug auf Prozesse unterstützt. Dies gilt auch für Ebene 5. Falls eine Organisation die Ebene 5 mit Hilfe des CMM-Ansatzes erreicht hat, können alle Grade von Verbesserungen unterstützt werden. Auch unterhalb dieser Ebene sind einmalige Verbesserungen erreichbar. Allerdings ist das Sichern von Wiederholbarkeit und Nachweisbarkeit von Verbesserungen kein expliziter Bestandteil des Ansatzes. Mittels des KPA „Quantitatives Prozeßmanagement“ können jedoch umgebungsabhängig wiederholbare und nachweisbare Verbesserungen erreicht werden.

Der AMI-Ansatz unterstützt teilweise den ingenieurwissenschaftlichen Ansatz, da von der Charakterisierung ausgehend die Ziele, die eine Verbesserung versprechen, definiert werden. Der empirische Ansatz wird nicht unterstützt. Zielorientiertes Messen ist Bestandteil des Ansatzes, nicht aber explizite Modellierung und Wiederverwendung. Der AMI-Ansatz ermöglicht genauso wie der Ad-hoc-Ansatz einmalige Verbesserungen und kann bei richtiger Anwendung des GQM-Ansatzes zu wiederholbaren Verbesserungen führen. Nachweisbare Verbesserungen sind durch den AMI-Ansatz nicht unterstützt, da sie explizite Modellierung erfordern. Dies ist jedoch kein Bestandteil des AMI-Ansatzes

Beim QIP werden die ingenieurwissenschaftliche und die empirische Vorgehensweise von Beginn an kombiniert angewendet, wie es den in Kap.2.1 beschriebenen Bedürfnissen der Software-Entwicklung entspricht. Es werden Modelle von Prozessen und Produkten gebildet, Projekte nach diesen Modellen durchgeführt, gemessen und analysiert. Um neue Methoden ohne große Risiken zu erproben, werden zusätzlich zu den Projekten Experimente durchgeführt. Die verschiedenen Modelle werden nach ihrer Verwendung in einem Projekt oder einem Experiment zur Wiederverwendung abgelegt. Messen und Modellieren sind integraler Bestandteil des QIP, siehe hierzu auch Kap. 3. Mittels QIP sind alle Grade von Verbesserungen erreichbar.

Die folgende Tabelle 3 faßt die Bewertung zusammen. Der Eintrag eines Minus (-) bedeutet keine Unterstützung des Aspektes, eine Null (0) bedeutet eine teilweise Unterstützung und ein Plus (+) eine volle Unterstützung.

Tabelle 3. Gegenüberstellung der verschiedenen Verbesserungsansätze

	Ad hoc	CMM		AMI	QIP
		E. 1-4	E. 5		
ingenieurwissenschaftliche Vorgehensweise	–	–	+	0	+
empirische Vorgehensweise	–	–	–	–	+
Messen	0	0	+	+	+
Modellieren	–	0	+	–	+
Wiederverwenden	–	0	0	–	+
einmalige Verbesserungen	0	+	+	+	+
wiederholbare Verbesserungen	–	0	+	0	+
nachweisbare Verbesserungen	–	0	+	–	+

3 Infrastrukturtechnologien für kontinuierliche Verbesserung nach dem QIP

Kap. 2 stellt das Quality Improvement Paradigm als den effektivsten Ansatz für kontinuierliche Qualitätsverbesserung vor. Der Ansatz stützt sich auf die drei Infrastrukturtechnologien zielorientiertes Messen, explizites Modellieren und umfassende Wiederverwendung (vergleiche Kap. 2.3), um kontinuierliche Qualitätsverbesserung zu ermöglichen. Alle drei Infrastrukturtechnologien sind notwendig, wir werden uns in diesem Kapitel jedoch auf zielorientiertes Messen konzentrieren. Kap. 3.1 beschreibt den GQM-Ansatz, weitere zielori-

entierte Meßansätze werden kurz vorgestellt und im Vergleich mit dem GQM-Ansatz bewertet. Explizites Modellieren und umfassende Wiederverwendung werden in Kap. 3.2 und Kap. 3.3 kurz umrissen.

3.1 Zielorientiertes Messen

Das Verständnis für ablaufende Software-Entwicklungsprozesse kann durch den Aufbau von Software-Entwicklungs-Know-how in der Form von Modellen verbessert werden. Die Modelle sollen die Realität möglichst genau abbilden. Sie werden mit Hilfe von quantitativen Daten aufgebaut und können ständig auf ihre Übereinstimmung mit der Realität geprüft werden. Die kontrollierte Durchführung von Projekten wird durch die entwickelten Modelle unterstützt und der Nachweis von Verbesserungen wird möglich.
Messungen unterstützen den Aufbau und die Validierung von Produkt- und Prozeßmodellen. Daher muß ein Meßansatz zur Unterstützung der kontinuierlichen Qualitätsverbesserung das Messen von Produkten und Prozessen unterstützen. Weitere Anforderungen des QIP an Meßansätze sind Zielorientierung, Zweckgebundenheit und Kontextabhängigkeit. Zielorientierung dient dazu, daß nur in Abhängigkeit eines bestimmten Zieles Maße definiert und Daten erhoben werden. Damit ist gesichert, daß keine überflüssigen Daten gesammelt werden, die von niemandem wirklich genutzt werden. Abhängig vom Zweck, zu dem Daten erhoben werden (verbessertes Verständnis, Kontrolle, Verbesserung) können unterschiedliche Maße interessant sein. Deswegen wird eine Orientierung der Messung an ihrem Zweck gefordert. Dies gilt auch für den Kontext, in dem die Messungen stattfinden. In verschiedenen Umgebungen gelten völlig unterschiedliche Anforderungen z.B. an die Zuverlässigkeit. Daher haben auch die Maße für Zuverlässigkeit nur kontextgebunden Gültigkeit. Bei der Definition von Messungen muß die Abhängigkeit vom Kontext berücksichtigt werden.
In der Literatur werden verschiedene Top-Down-Ansätze beschrieben: Der Quality Function Deployment Ansatz (QFD), der Software Quality Metrics Ansatz (SQM) und der Goal/Question/Metric Ansatz (GQM). In diesem Abschnitt konzentrieren wir uns auf GQM-Ansatz. Der SQM- und der QFD-Ansatz werden kurz beschrieben, und alle drei Ansätze werden bezüglich ihrer Eignung im Rahmen kontinuierlicher Qualitätsverbesserung verglichen.

3.1.1 Der GQM-Ansatz

Der GQM-Ansatz wurde von Victor Basili, Dieter Rombach und anderen an der University of Maryland entwickelt [BaWe84, BaRo88, Romb91, Basi92, BaCR94] und wird zur Zeit an der University of Maryland und an der Universität Kaiserslautern weiterentwickelt.
Dieser Abschnitt beschreibt verschiedene Aspekte des GQM-Ansatzes. Zunächst werden die Prinzipien des Ansatzes genannt. Dann wird auf die Methode, die zur Anwendung des An-

satzes verfolgt wird, eingegangen. In einem letzten Abschnitt werden Techniken und Werkzeuge beschrieben, die den GQM-Ansatz unterstützen.

3.1.1.1 Prinzipien des GQM-Ansatzes

Die maßgeblichen Prinzipien des GQM-Ansatzes sind:

- Festlegung expliziter Ziele. Bevor mit der Definition von Messungen begonnen wird, werden die Ziele der Messung explizit festgelegt. Meßziele sind immer Analyseziele. Es wird bestimmt, was gemessen wird und wozu, auf welche Eigenschaften sich die Messungen konzentrieren, von welchem Blickwinkel aus gemessen wird und in welcher Umgebung.
- Top-down-Definition/Bottom-up-Interpretation. Vom Ziel ausgehend werden die Maße definiert, die erforderlich sind, um das Ziel zu erfüllen. Wenn die Daten erfaßt sind, werden sie im Kontext des Zieles interpretiert.
- Definition und Interpretation in enger Zusammenarbeit mit den beteiligten Projektmitgliedern. Zur Ableitung der Maße aus dem Ziel werden die Personen befragt, aus deren Blickwinkel heraus das Ziel definiert wurde. Damit ist sichergestellt, daß diejenigen Daten erfaßt werden, die von diesen Personen benötigt werden. Auch die Interpretation der erfaßten Daten wird im Kontext des Zieles und von den Personen geleistet, die am Definieren der Maße oder am Liefern der Daten beteiligt sind. Falsche Schlußfolgerungen, die aus Unkenntnis der tatsächlichen Gegebenheiten gezogen werden könnten, werden vermieden.

3.1.1.2 GQM-Methode

Die im vorherigen Abschnitt beschriebenen Prinzipien werden durch die GQM-Methode umgesetzt [GrHW95]. Die GQM-Methode unterstützt die Anwendung des GQM-Ansatzes. Sie umfaßt die Planung und Durchführung von Messungen im Kontext eines Meßprogrammes für ein bestimmtes Projekt. Ein Meßprogramm beinhaltet alle Meßaktivitäten eines Projektes. Bevor auf die einzelnen Aktivitäten bei einem Meßprogramm nach der GQM-Methode näher eingegangen wird, werden die verschiedenen Dokumente, die bei der GQM-Methode erstellt werden, kurz vorgestellt.

- Das zentrale Dokument der Methode ist der GQM-Plan, in neueren Publikationen auch GQM-Modell genannt [BaCR94]. Ein GQM-Plan enthält eine Menge von Informationen. Sie dokumentieren, mit welcher Motivation Maße aus einem bestimmten Ziel abgeleitet werden, welche Maße das sind und wie die gesammelten Meßdaten im Kontext des Zieles interpretiert werden.
- Zu einem GQM-Plan gehören eine oder mehrere GQM-Masken. Eine GQM-Maske enthält eine Abstraktion der Informationen, die in einem GQM-Plan enthalten sind. GQM-

Masken sind wesentliche Hilfsmittel zur expliziten Beteiligung der Projektmitglieder. Sie werden verwendet, um die von Projektmitgliedern erhaltenen Informationen zu strukturieren und um GQM-Pläne leicht lesbar darzustellen.

- Ein Meßplan enthält Informationen, die den Prozeß der Datensammlung für ein Projekt beschreiben. Im Meßplan wird festgehalten, welche Daten wann, durch wen und wie erhoben werden. In einem Meßplan können die Maße mehrerer GQM-Pläne integriert werden. Außerdem enthält er Informationen darüber, wer die Daten validiert und wer für die Datenhaltung zuständig ist.

Die GQM-Methode zur Unterstützung des GQM-Ansatzes ist analog zum QIP in sechs Schritte aufgeteilt. Schritt 1, 2 und 3 beinhalten die Planung der Messung, Schritt 4 die projektbegleitende Durchführung, Schritt 5 die post-mortem-Analyse der Daten und Schritt 6 die Sicherung der gewonnenen Erfahrung. Hierbei wird auf die projektbezogene Sicht des QIP Bezug genommen (vergleiche Tabelle 2.3).

Schritt 1: Charakterisiere
Schritt 2: Identifiziere GQM-Ziele
Entwickle GQM-Pläne
Schritt 3: Entwickle Meßplan
Schritt 4: Sammle und validiere Daten
Analysiere und interpretiere Daten projektbegleitend
Schritt 5: Analysiere und interpretiere Daten post-mortem
Schritt 6: Sichere Ergebnisse

Schritt 1: Charakterisiere. In Schritt 1 wird innerhalb des Projekts, in dem ein GQM-basiertes Meßprogramm durchgeführt werden soll, bereits existierendes und relevantes Know-how und nützliche Eingangsdokumente, z.B. Projektbeschreibungen o.ä., identifiziert. Das Anwendungsprojekt wird charakterisiert.

Schritt 2a: Identifiziere GQM-Ziele. In diesem Schritt werden ausgehend von Organisations- und Projektzielen Meßziele hergeleitet. Diese Meßziele bilden die Grundlage der Entwicklung des Meßprogramms. GQM-Ziele haben eine ganz bestimmte Form, die im nächsten Abschnitt genauer erläutert wird. Die Meßziele werden als GQM-Ziele formuliert und sehen dann z.B. folgendermaßen aus: Analysiere den Komponententest zum Zwecke des Verstehens seiner Effektivität aus dem Blickwinkel des Testers in der Umgebung Projekt A in Firma B.

Schritt 2b: Entwickle GQM-Pläne. Für jedes der GQM-Ziele des Meßprogramms wird ein GQM-Plan entwickelt. Die Entwicklung eines GQM-Plans gliedert sich in verschiedene Aktivitäten: Es werden Interviews mit Personen durchgeführt, die den im Ziel enthaltenen Blickwinkel repräsentieren, im o.g. Beispielziel die Tester. Diese Interviews sichern die intensive Einbeziehung der Projektmitglieder in den Prozeß des Datendefinition. Dies ist eines

der Prinzipien des GQM-basierten Messens. Die Ergebnisse der Interviews werden in GQM-Masken festgehalten. Diese GQM-Masken enthalten die Sicht der Projektmitglieder auf das zu erreichende Ziel. Die erhaltenen Informationen werden im entwickelten GQM-Plan detailliert und strukturiert. Meist sind mehrere Iterationen von Interview und GQM-Plan-Entwicklung erforderlich, bis der GQM-Plan alle Informationen enthält, die zur Messung erforderlich sind. Die Inhalte des fertiggestellten GQM-Plan werden abstrahiert in einer GQM-Maske festgehalten. Diese abstrakte Sicht wird in einem Review mit den Personen besprochen, die interviewt wurden. Ziel dieses Reviews ist die Feststellung, ob die Sicht der interviewten Personen in den entwickelten GQM-Plänen reflektiert wurde. Ein Beispiel einer GQM-Maske und ein Auszug des entsprechenden GQM-Plans ist in Kap. 3.1.1.3 enthalten.

Schritt 3: Entwickle Meßplan. Es wird ein Meßplan entwickelt, der die Maße der verschiedenen GQM-Pläne des Meßprogramms integriert. Für jedes Maß wird definiert, mit welchem Mechanismus es erhoben wird (Werkzeug, Fragebogen) und von welchen Personen oder welchen Werkzeugen die Daten zu welchem Zeitpunkt geliefert werden können. Die definierten Messungen müssen mit dem ablaufenden Prozeß in Einklang stehen. Daher ist es erforderlich, sich bei der Entwicklung des Meßplans auf die Prozeßmodelle zu stützen, die den ablaufenden Software-Entwicklungsprozeß beschreiben, vergleiche [BrDT96].

Schritt 4a: Sammle und validiere Daten. Begleitend zum Projekt werden die Maße zu dem im Meßplan vorgegebenen Zeitpunkten erhoben, validiert und abgelegt.

Schritt 4b: Analysiere und interpretiere Daten projektbegleitend. Während der Dauer des Projektes werden periodisch (z.B. alle vier oder sechs Wochen) Sitzungen zur Interpretation der Daten abgehalten. Diese Sitzungen werden Feedback-Sitzungen genannt. Die gesammelten und aufbereiteten Daten werden dort präsentiert und im Kontext des jeweiligen Zieles diskutiert. An den Feedback-Sitzungen nehmen die Personen teil, die bei der Ableitung der Daten mittels Interviews beteiligt waren, und solche, die Daten geliefert haben. Die Personen, die bei den Interviews befragt wurden, interpretieren während der Feedback-Sitzungen die aufbereiteten Meßdaten. Die Personen, die Daten geliefert haben, informieren sich in diesen Sitzungen über die Nutzung der Daten und können in der Diskussion Einfluß auf die Interpretation der Daten nehmen. Die Ergebnisse der Feedback-Sitzungen werden zur Projektkontrolle und zur Qualitätsverbesserung verwendet.

Schritt 5: Analysiere und interpretiere Daten post-mortem. Die Daten, die während des Projektes angefallen sind, werden kontextbezogen innerhalb der GQM-Pläne analysiert. In diesem Schritt werden Fragestellungen werden analysiert und diskutiert, die über das einzelne Projekt hinausgehen.

Schritt 6: Sichere Ergebnisse. Alle zu diesem Meßprogramm gehörenden Dokumente und Daten sowie die gesammelten Erfahrungen werden strukturiert und mit Kontextinformationen versehen. So aufbereitet werden sie in der Erfahrungsdatenbank abgelegt.

3.1.1.3 Techniken und Werkzeuge

In diesem Abschnitt werden Techniken und Werkzeuge vorgestellt, die den GQM-Ansatz unterstützen. Techniken sind Hilfsmittel zur Unterstützung einer Vorgehensweise (z.B. Formalismen), Werkzeuge sind rechnergestützte Techniken.

Techniken

Bei der Durchführung eines Meßprogramms nach der GQM-Methode werden unterschiedliche Dokumente erstellt: GQM-Pläne, GQM-Masken und ein Meßplan. Die GQM-Pläne enthalten die Informationen, die die Messungen motivieren und die Datenverwendung definieren, GQM-Masken fassen diese Informationen zusammen. Der Meßplan unterstützt die Datensammlung und die Datenvalidierung. Sein Inhalt ist eher technischer Natur und wird hier nicht genauer beschrieben.

Im folgenden wird detaillierter auf die Struktur von GQM-Plänen eingegangen. Ein GQM-Plan besteht aus einem Ziel, einer Menge von Fragen und einer Menge von Maßen. Sie sind auf verschiedenen Ebenen angeordnet: Das Ziel auf der konzeptuellen Ebene, die Fragen auf der operationalen Ebene und die Maße auf der quantitativen Ebene. Die Fragen sind innerhalb der operationalen Ebene zusätzlich in Gruppen eingeteilt:

- **Die konzeptuelle Ebene**: Ein Meßziel ist definiert durch die Angabe eines Objektes, eines Zwecks, eines Qualitätsmerkmals, eines Blickwinkels und eines Kontextes. Objekte der Messung können Prozesse, Produkte und Ressourcen (Werkzeuge, Personal, etc.) sein. Die Messung kann Zwecken unterschiedlicher Art dienen, z.B. Charakterisierung, Kontrolle, Verbesserung. Beliebige Objekte können analysiert werden mit Konzentration auf unterschiedliche Qualitätsaspekte und aus verschiedenen Blickwinkeln heraus. GQM-Ziele werden wie folgt formuliert [Romb91]:

Analysiere	den Komponententest	[Objekt]
zum Zwecke	des Verstehens	[Zweck]
bezüglich	der Effektivität	[Qualitätsaspekt]
aus dem Blickwinkel	des Testers	[Blickwinkel]
bei	Projekt A der Firma B	[Kontext]

- **Die operationale Ebene**: Das Meßziel wird auf der operationalen Ebene durch eine Menge von Fragen charakterisiert. Die Fragen halten Eigenschaften des zu messenden Objektes fest, die im Kontext des Zieles interessant sind. Auf der Ebene der Fragen existiert eine weitere Unterteilung in drei verschiedene Fragengruppen, die Subziele genannt werden. Eine der drei Gruppen, Qualität genannt, beschäftigt sich mit den Fragen, die das Qualitätsmerkmal genauer beschreiben. Ein Beispiel für eine Frage dieser Gruppe im Kontext des o.g. Beispielzieles ist: „Wieviele Fehlverhalten werden pro Testdurchlauf gefunden?“ Eine weitere Gruppe von Fragen, Prozeß- bzw. Produktdefinition genannt, beschäftigt sich mit den Merkmalen von Produkten oder Prozessen, die einen Einfluß auf

Teile des Qualitätsmerkmals haben. Ein Beispiel für eine Frage dieser Gruppe ist: „Welche Teststrategie wurde gewählt?“ Eine dritte Gruppe von Fragen beschäftigen sich mit dem Feedback, den die verfolgte Analyse auf Projekte oder auf bisher angesammeltes Know-how haben kann. Diese drei Gruppen von Fragen können weiter unterteilt werden. Es werden zwei verschiedene Strukturen für die Gruppierung von Fragen vorgeschlagen, je nachdem, ob der GQM-Plan Produkte oder Prozesse analysiert. Die Struktur von GQM-Plänen, die Prozesse analysieren, ist in Abb 6 enthalten, die Struktur für solche Pläne, die Produkte analysieren, in Abb 7. Für eine detailliertere Beschreibung dieser Fragengruppen vergleiche [Romb91].

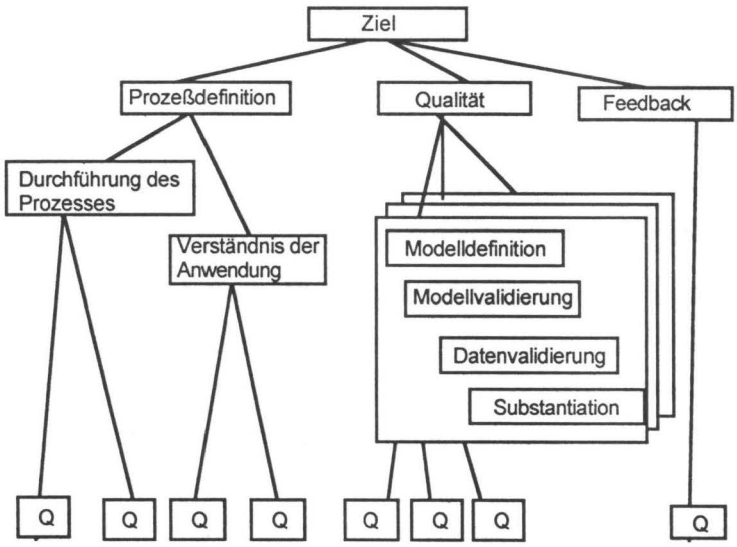


Abb. 6. Struktur für Prozeß-GQM-Pläne

– **Die quantitative Ebene**: Eine Menge von Maßen beantwortet jede Frage auf eine quantitative Art. Beispielmaße für die Frage „Wieviele Fehlverhalten werden pro Testdurchlauf gefunden?“ sind „Anzahl der Testdurchläufe" und „Anzahl der gefundenen Fehlverhalten“. Durch die Maße wird festgelegt, welche Daten erhoben werden, um die Fragen zu beantworten.

Zu jedem GQM-Plan gehören eine oder mehrere GQM-Masken. Sie werden als Strukturierungshilfe für Interviews und für Reviews der GQM-Pläne verwendet. Abb. 8 enthält die Struktur der GQM-Maske. Bei den Interviews werden leere oder bereits teilweise ausgefüllte Masken benutzt, um die im Interview besprochenen Merkmale an der richtigen Stelle festzuhalten.

Wenn der GQM-Plan fertiggestellt ist, wird eine GQM-Maske, die eine abstrakte Darstellung dieses GQM-Plans enthält, entwickelt. Abb. 9 enthält ein Beispiel für eine solche GQM-Maske. Die Subziele der GQM-Pläne sind in der GQM-Maske als Quadranten repräsentiert. Während des Interviews können so die zur Sprache kommenden Aspekte sofort eingeordnet

werden. Merkmale zur Beschreibung des Qualitätsaspektes (Subziel Qualität) werden im 1. Quadranten festhalten. Die Einflußfaktoren auf den Qualitätsaspekt, im GQM-Plan unter dem Subziel Prozeß- oder Produktdefinition enthalten, werden im 2. Quadranten abgelegt. Informationen zum Feedback sind im Feedback-Teil unten auf der GQM-Maske enthalten.

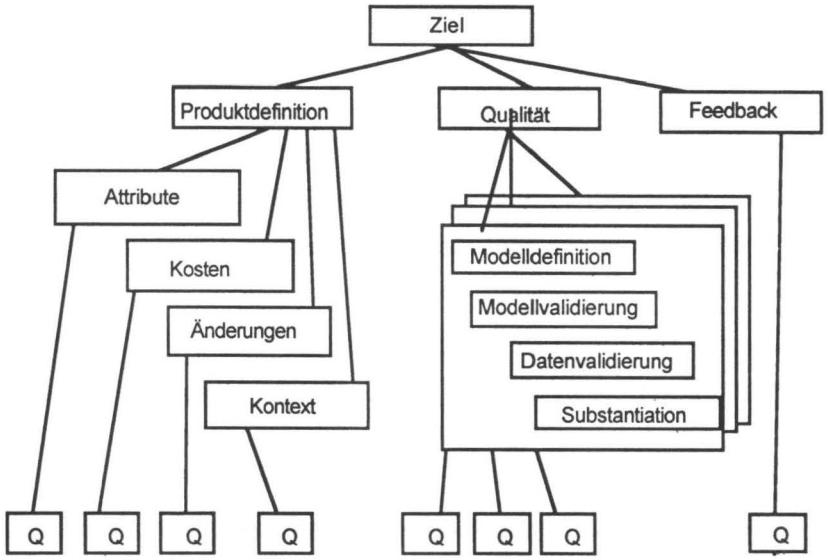


Abb. 7. Struktur für Produkt-GQM-Pläne

Zusätzlich zu diesen Informationen, die der Definition der Messungen dienen, sind im 3. und 4. Quadranten Informationen enthalten, die für die Auswertung der gesammelten Daten benötigt werden. Der 3. Quadrant enthält eine Quantifizierung der im 1. Quadranten angegebenen Merkmale des Qualitätsaspektes, vergleiche Beispiel in Abb. 9. Bei der Interpretation werden die Quantifizierungen zum Soll/Ist-Vergleich genutzt. Diese Quantifizierung beruht entweder auf Schätzungen der interviewten Personen oder auf bereits vorhandenen Meßdaten. Die Informationen, die im 4. Quadranten enthalten sind, setzen jeden der Einflußfaktoren aus dem 2. Quadranten in Beziehung zu einem der Merkmale im1. Quadranten, z.B. „Je komplexer der Entwurf, um so höher der Aufwand für die Phase Testen". Im GQM-Plan wird die Beschreibung des vermuteten Einflusses im Definitionssubziel festgehalten, bei der Frage nach der Komplexität des Entwurfs. Abb. 9 zeigt Auszüge einer GQM-Maske, Abb. 10 den entsprechenden GQM-Plan.

Objekt Zweck Qualitätsaspekt Blickwinkel Kontext	
Qualitätsaspekt 1. Quadrant	Einflußfaktoren 2. Quadrant
Quantifizierung des Qualitätsaspektes 3. Quadrant	Wirkung der Einflußfaktoren auf den Qualitätsaspekt 4. Quadrant
Feedback:	

Abb. 8. Struktur der GQM-Maske

Werkzeuge

Zur Unterstützung der Erstellung von GQM-Plänen wurde GQM-DIVA entwickelt, ein graphischer GQM-Plan-Editor [VanM95]. Dieses Prototyp-Werkzeug basiert auf einer formalen GQM-Sprache, die die geschilderte Struktur von GQM-Plänen widerspiegelt [Diff93].

Eine detailliertere Beschreibung der GQM-Methode enthält [GrHW95]. [BiKe95a] und [BiKe95b] enthalten Tutorien über den GQM-Ansatz für verschiedene Lernziele.

3.1.2 Weitere Ansätze zum zielorientierten Messen

Neben dem GQM-Ansatz existieren noch weitere zielorientierte Meßansätze, die bei der Software-Entwicklung eingesetzt werden. In diesem Abschnitt werden der QFD-Ansatz und der SQM-Ansatz vorgestellt.

3.1.2.1 Der QFD-Ansatz

Der QFD-Ansatz wurde von Yoji Akao 1966 an der Tamagawa University of Tokyo entwickelt [KoAk83]. Ursprünglich wurde dieser Ansatz verwendet, um Planungsaktivitäten für industrielle Produktionsprozesse zu unterstützen. Später wurde er für die Unterstützung bei der Software-Entwicklung angepaßt. Der Ansatz stellt einen Rahmen bereit, in dem Qualitätsaspekte des Endprodukts bei der Planung betrachtet werden können. Ein Produkt wird aus der Sicht der Kunden und der Benutzer betrachtet, um diejenigen Aspekte des Produktes und des Prozesses zu ermitteln, die bei der Entwicklung besonders beachtet werden müssen.

Dem QFD-Ansatz liegt das Prinzip zugrunde, daß Charakteristika des zu erzeugenden Endproduktes sich auf verwandte Charakteristika von Zwischenprodukten wie z.B. Anforderungen oder Entwurf beziehen. So können die Charakteristika der Zwischenprodukte zur Vorhersage der relevanten Eigenschaften des Endproduktes verwendet werden. Beispielsweise wird ein Entwurfsdokument analysiert, um Vorhersagen bezüglich der Wartbarkeit des Endproduktes zu treffen. Der QFD-Ansatz stellt keine Anleitung für die Ableitung von interessierenden Charakteristika und Maßen aus den Zielen bereit.

Objekt Komponententest	**Zweck** Verstehen	**Qualitätsaspekt** Effektivität	**Blickwinkel** Tester	**Umgebung** Firma X

Beschreibung des Qualitätsaspektes Effizienz

1 Anzahl der entdeckten Fehlverhalten

2 Kritische und unkritische Fehlverhalten

3 Anzahl der entdeckten Fehler im Test

4 Verteilung auf drei Fehlerklassen

. . .

Einflußfaktoren

1 Entwicklungsdauer für Testfälle

2 Qualität der Testdaten

3 Erfahrung der Tester mit Testwerkzeugen

4 Vorgegebene Teststrategie

5 Befolgung der Teststrategie

6 Verständnis der Anforderungen

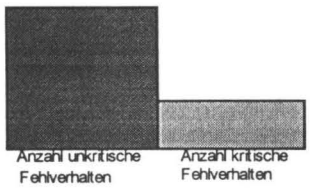
7 Verständnis des Quellprogramms

. . .

Quantifizierung des Qualitätsaspektes

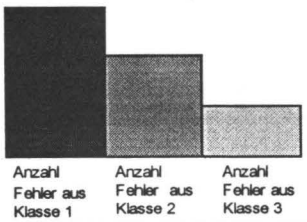
1 20

2



3 30

4



Wirkung der Einflußfaktoren

1 Je länger Entw.dauer f.Testfälle, um so höher die Anzahl der entdeckten Fehlverhalten

2 Je höher d. Qualität d. Testdaten, um so höher die Anzahl der entdeckten Fehlverhalten

3 Je höher die Erfahrung d. Tester mit den Testwerkzeugen, umso höher die Anzahl der entdeckten Fehlverhalten

4 Je nach Teststrategie ändert sich die Anzahl der entdeckten Fehlverhalten

5 Nichtbefolgen d. Teststrategie vermindert die Anzahl der entdeckten Fehlverhalten

6 Je geringer d. Verständnis der Anforderungen, um so geringer die Anzahl der entdeckten Fehlverhalten

7 Je geringer das Verständnis des Quellprogramms,umso geringer die Anzahl der entdeckten Fehler

Feedback

Abb. 9. Beispiel für eine ausgefüllte GQM-Maske

Analysiere	den Komponententest	[Objekt]
Zum Zwecke	des Verstehens	[Zweck]
Bezüglich	der Effektivität	[Qualitätsaspekt]
Aus dem Blickwinkel	des Testers	[Blickwinkel]
Bei	Projekt A der Firma B	[Kontext]

Prozeßdefinition
Qualität der Durchführung

Frage D.1: Wieviel Zeit wurde zur Definition der Testfälle benötigt?
Maß MD.1.1: Aufwand_für_Testfälle
Frage D.2: Wieviel Zeit wurde zur Definition der Testdaten benötigt?
Maß MD.2.2: Aufwand_für_Testdaten
Frage D.3: Wieviel Erfahrung hatten die Tester mit den benutzten Werkzeugen?
Maß MD.3.1: Erfahrung_der_Tester
Frage D.4: Welche Teststrategie und Abbruchkriterien wurden vorgegeben?
Maß MD.4.1: Liste_Teststrategie_und_Abbruchkriterien
Frage D.5: Wie gut wurde die vorgegebene Testmethode befolgt?
Maß MD.5.1: Befolgung_der_Teststrategie
Frage D.6: Wie gut verstehen die Tester die Anforderungen?
Maß MD.6.1: Verständnis_Anforderungen_der_Tester
Frage D.7: Wie gut verstehen die Tester das Quellprogramm?
Maß MD.7.1: Verständnis_Quellprogramm_der_Tester

Qualität Effektivitätmodell
Modelldefinition

Frage Q1: Wieviele Fehlverhalten wurden entdeckt?
Maß MQ.1.1: Anzahl_entdeckter_Fehlverhalten
Frage Q2: Wie ist die Anzahl kritischer und unkritischer Fehlverhalten?
Maß MQ.2.1: Anzahl_entdeckter_kritischer_Fehlverhalten
Maß MQ.2.2: Anzahl_entdeckter_unkritischer_Fehlverhalten
Frage Q3: Wieviele Fehler wurden entdeckt?
Maß MQ.3.1: Anzahl_entdeckter_Fehler
Frage Q4: Wie ist die Anzahl verteilt auf die einzelnen Fehlerklassen?
Maß MQ.4.1: Anzahl_entdeckter_Fehler_der_Klasse_1
Maß MQ.4.2: Anzahl_entdeckter_Fehler_der_Klasse_2
Maß MQ.4.3: Anzahl_entdeckter_Fehler_der_Klasse_3

Abb. 10. Beispiel für einen GQM-Plan

3.1.2.2 Der SQM-Ansatz

Der SQM-Ansatz wurde 1979 von Gerald E. Murine entwickelt [Muri80]. Er basiert auf einem Ansatz zum Quantifizieren von Software-Charakteristika von Boehm [BoBL76], der von McCall [MaRW77] weiterentwickelt wurde. Die Ziele der Messungen konzentrieren sich beim SQM-Ansatz auf das Endprodukt aus der Sicht des Kunden oder Benutzers. Ihr Zweck ist die Bewertung und Zertifizierung. Die Anwendung des Ansatzes wird auf diejenigen Qualitätsfaktoren beschränkt, die für die Zielgruppe der Kunden wichtig sind. Eine vordefinierte Hierarchie aus Faktoren (z.B. Wartbarkeit), Kriterien (z.B. Strukturiertheit als ein Kriterium für den Faktor Wartbarkeit) und Maßen (z.B. Coupling als ein Maß für das Kriterium Strukturiertheit) wird vorgeschlagen. Die für den konkreten Fall passenden Faktoren, Kriterien und Maße werden aus dieser vordefinierten Hierarchie ausgewählt. Es wird angenommen, daß alle Elemente, die in einem Projekt von Interesse sein können, in der Hierarchie vorhanden sind. Es wird keine Unterstützung bereitgestellt, die vordefinierte Hierarchie von Faktoren, Kriterien und Maße zu ändern.

3.1.2.3 Vergleich der Ansätze

Die drei vorgestellten Meßansätze werden verglichen in Bezug auf die Erfüllung der Anforderungen, die innerhalb des QIP an Meßansätze gestellt werden: Zielorientierung, Zweckgebundenheit, Kontextabhängigkeit und Betrachtung von Prozeß- und Produktmaßen, vergleiche Kap. 3.1. Weiterhin wird betrachtet, welche Unterstützung die Ansätze bei der Ableitung von Maßen aus Zielen bieten, da dies wesentlich ihre praktische Anwendung unterstützt, vergleiche [Romb91].

Der QFD-Ansatz ist zielorientiert und zweckgebunden, wobei sich der Zweck auf die Sicherung von bestimmten Qualitätsmerkmalen des Endproduktes beschränkt ist. Insgesamt werden nur Produktmerkmale betrachtet. Prozeßmerkmale und Merkmale, die den Kontext betreffen, werden vernachlässigt. Ein weiterer Schwachpunkt des Ansatzes ist der Mangel an Hilfestellung zur Ableitung von Charakteristika und Maßen.

Auch der SQM-Ansatz ist zielorientiert, der Zweck beschränkt sich jedoch auf Bewertung und Zertifizierung. Es werden nur Produkt- und keine Prozeßmaße betrachtet. Die Hierarchie von Faktoren, Kriterien und Maßnahmen ist vordefiniert und es existieren keine Richtlinien zur Erweiterung.

Tabelle 4. Vergleich der verschiedenen Meßansätze

Kriterien	QFD	SQM	GQM
Anforderungen des QIP:			
Zielorientierung	ja	ja	ja
Zweckgebundenheit	eingeschränkt	eingeschränkt	ja
Kontextabhängigkeit	nein	nein	ja
Prozeßmaße	nein	nein	ja
Produktmaße	ja	ja	ja
Ableitung von Maßen:			
Prinzip	Ableitung von Produktchar. an verschiedenen Stellen des Entwicklungsstadiums aus Benutzerchar. des auszuliefernden Produkts	Verfeinere Faktoren in Kriterien und Maße	Verfeinere Ziele in Fragen und Maße
Mechanismus	auswählen, anpassen	auswählen	auswählen, anpassen

Bei Anwendung des GQM-Ansatzes können beliebige Ziele zu beliebigen Zwecken und aus beliebigen Blickwinkeln definiert werden. Die Kontext wird bei diesem Ansatz mitbetrachtet. Außerdem bietet der GQM-Ansatz flexible Mechanismen zur Ableitung von Maßen an, sei es bezogen auf Produkte oder auf Prozesse. Der GQM-Ansatz kann zur Unterstützung aller im QIP geforderten Meßaktivitäten verwendet werden und ist zur Unterstützung kontinuierlicher Qualitätsverbesserung geeignet.

3.2 Explizites Modellieren

Eine weitere Infrastruktur des QIP ist das explizite Modellieren von Software-Produkten und Prozessen. Explizite Prozeßmodelle dokumentieren die bei der Entwicklung gewählte Vorgehensweise, sie unterstützen die Planung von Projekten, die Wiederverwendung von Prozessen und das Einpassen neuer Techniken in einen existierenden Prozeß. Ein weiterer Vorteil ist die Nachweisbarkeit gewählter Vorgehensweisen durch Prozeßmodelle und damit die Möglichkeit, Qualität der Software-Entwicklung zu dokumentieren. Angereichert mit Meßdaten, die modellierte Eigenschaften von Prozessen und Produkten betreffen, liefern Prozeßmodelle eine nachvollziehbare und analysierbare Beschreibung eines abgelaufenen Entwicklungsprojektes.

Eine Prozeßmodellierungssprache zur Unterstützung des QIP muß folgenden Anforderungen genügen:

- Alle relevanten Objekte (Produkte, Prozesse, Ressourcen, Eigenschaften,..) sind modellierbar;
- Alle wichtigen Beziehungen zwischen diesen Objekten (Produktfluß, Kontrollfluß, Ressourcenzuordnung,...) werden formuliert;

- Es werden geeignete Modularisierungskonzepte bereitgestellt (Verfeinerung / Aggregation, Sichtenextraktion /-kombination,...);
- Die Modelle werden innerhalb eines Projektplans von Menschen oder automatisch ausgeführt;
- Die Projektpläne sind für Messungen instrumentierbar;
- Der Projektverlauf kann protokolliert werden;
- Projektpläne können analysiert werden;
- Zur Ermöglichung von Wiederverwendung werden geeignete Konzepte zur Ablage und Kombination von Modellen bereitgestellt (Spezialisierung, Generalisierung).

Die Prozeßmodellierungssprache MVP-L (Multi View Process Modeling-Language) [BLRV95], die an der Universität Kaiserslautern entwickelt wurde, wird den oben angeführten Anforderungen größtenteils gerecht. Im folgenden beschreiben wir die Aspekte von MVP-L, die für das QIP und damit für die Zusammenarbeit der verschiedenen Infrastrukturtechnologien besonders relevant sind.

Die Hauptelemente von MVP-L sind Modelle von Prozessen, Produkten und Ressourcen sowie deren Eigenschaften, Attribute genannt. Die Modelle können mittels eines Projektplans kombiniert und instanziiert werden. Diese umfassende Prozeßbeschreibung wird Projektplan genannt und ist ausführbar. Prozesse und Produkte können in Aggregationshierarchien angeordnet werden. Die Extraktion von verschiedenen Sichten auf einen Projektplan wird unterstützt.

Durch die Attribute der verschiedenen Modelle erfolgt die Integration mit den Messungen. Die Messungen halten zu bestimmten Zeitpunkten die Eigenschaften von Prozessen und Produkten als Attributwerte fest. Diese Attributwerte steuern den Kontrollfluß innerhalb von Eingangs-/Ausgangsbedingungen. Durch diese Bedingungen wird nicht ein einziger Kontrollfluß beschrieben, sondern die Menge aller möglichen Kontrollflüsse. Durch die Eingangs-/Ausgangsbedingungen werden nur solche Wege ausgeschlossen, die sich bereits als nicht sinnvoll gezeigt haben. Mit wachsender Erfahrung können so immer mehr sinnvolle Einschränkungen in den Prozeßmodellen gemacht werden. Eine wichtige Hilfe beim Aufbau solcher Erfahrung sind Messungen. Durch die Analyse der Meßdaten können neue Erfahrungen zur Unterstützung der Vorhersage in die Prozeßmodelle eingebaut werden. So kann z.B. die geplante Testdauer einer Komponente abhängig von der Komplexität des Entwurfs dieser Komponente vorhergesagt werden. Dies ist kann berechnet werden, sobald der Entwurf abgeschlossen ist.

MVP-L kann sowohl für präskriptive als auch für deskriptive Modellierung verwendet werden. Die Sprache unterstützt die Erfassung des Ist-Prozesses eines Projektes, die Planung weiterer Projekte und die kontinuierliche Weiterentwicklung der Entwicklungsprozesse. MVP-L wurde bereits in mehreren Industrieprojekten erfolgreich eingesetzt.

3.3 Umfassende Wiederverwendung

Die dritte wichtige Infrastrukturtechnologie ist die umfassende Wiederverwendung. Innerhalb des QIP gelten ganz bestimmte Anforderungen an die Wiederverwendung: Sie darf sich nicht alleine auf den Code beschränken, es werden auch Zwischenprodukte wie Spezifikationen und Entwurf wiederverwendet, genauso wie Entwicklungsprozesse. Außerdem werden Qualitätsmodelle und andere Arten von Erfahrungen wiederverwendet. Produkte und Prozesse werden zusammen mit den für sie geltenden Kontextinformationen abgelegt. Wiederverwendung muß vollständig integriert sein in die Entwicklungsprozesse, die in der Unternehmung ablaufen. Sie beinhaltet fast immer eine Anpassung eines abgelegten Objektes an den neuen Kontext, innerhalb dessen es wiederverwendet werden soll.
Das Ziel von Entwicklungsprojekten ist die Auslieferung von Produkten. Deswegen werden Objekte in wiederverwendbarer Form nicht als Nebenprodukt innerhalb der Entwicklungsprojekte erzeugt. Diese Arbeit muß zusätzlich zur Entwicklungsarbeit geleistet werden. Sie findet statt in der „Experience Factory", die logisch und eventuell auch physisch getrennt von der Projektorganisation existiert. Neben dem Analysieren, dem Sichern und dem Strukturieren der Erfahrung ist das Anpassen der abgelegten Objekte an Projektgegebenheiten die Aufgabe der „Experience Factory". Die „Experience Factory"-Organisation muß in jedem Unternehmen passend zu den dort vorhandenen Bedürfnissen und Strukturen aufgebaut werden.

4 Start eines Qualitätsverbesserungsprogramms

Qualitätsverbesserungsprogramme nach dem QIP beginnen mit einem Pilotprojekt. Ein solches Pilotprojekt soll zu einer Menge für die Unternehmung typischer Projekte gehören und die Mitglieder sollen technologischen Neuerungen gegenüber positiv eingestellt sein. In diesem Projekt wird ein Meßprogramm nach der GQM-Methode gestartet, so wie sie in Kap. 3.1.1.2 beschrieben wurde. Mit Hilfe dieses Meßprogramms können erste quantitative Modelle der ablaufenden Prozesse und der erstellten Produkte gebildet werden. Das Meßprogramm faßt einige Meßziele zusammen, die unter anderem z.B. Aufwand und Zuverlässigkeit betreffen. Die resultierenden Modelle sind dann z.B. die prozentuale Verteilung des Aufwands, die Zuverlässigkeit der Zwischenprodukte und des Endprodukts.
Bei der Auswahl der Meßziele ist es wichtig, jeweils ein Meßziel für alle an dem Programm beteiligten Gruppen, wie z.B. Management, Qualitätssicherung und Entwickler, zu verfolgen. So wird gesichert, daß für alle Beteiligten das Meßprogramm einen Gewinn darstellt. Die Projektmitglieder müssen in allen Phasen des Meßprogramms (Planung, Durchführung, Auswertung) beteiligt werden. Sonst besteht die Gefahr, Modelle aufzubauen, die entweder den tatsächlichen Gegebenheiten nicht entsprechen oder nicht alle relevanten Faktoren beinhalten. Die entwickelten Modelle sind eine explizite, quantitative Dokumentation des Ist-Zustands des Projektes und des erzeugten Produktes. Wenn das Meßprogramm des Pilot-

projektes erfolgreich abgeschlossen wurde, ist der Ist-Zustand für dieses Projekt erfaßt. Ein erster Schritt zur Verbesserung des Know-hows der Organisation ist getan (siehe Kap. 2.2) Bezogen auf den Ist-Zustand des Pilotprojektes können jetzt Verbesserungsziele definiert werden. Gleichzeitig können Meßprogramme für weitere Projekte geplant werden. Spätestens zu diesem Zeitpunkt muß der Kern der „Experience Factory" aufgebaut werden. Diese Modelle des Ist-Zustandes des Pilotprojektes stellen die ersten Elemente in der Erfahrungsdatenbank der „Experience Factory" dar. Die im Pilotprojekt gewonnene Erfahrung muß gesichert werden, damit sie für weitere Projekte zur Verfügung gestellt werden kann. Meßprogramme werden von Mitarbeiterinnen der „Experience Factory" in enger Zusammenarbeit mit den Projektmitgliedern entwickelt, das gleiche gilt für Prozeß- und Produktmodelle. Die Analyse der in der Erfahrungsdatenbank abgelegten Modelle findet in der „Experience Factory" statt. Wenn die organisatorische Struktur einer „Experience Factory" angelegt ist, können Meßprogramme in weiteren Projekten durchgeführt werden. Dabei können die Erfahrungen aus dem ersten Projekt bereits wiederverwendet werden.
Beim Aufbau eines Qualitätsverbesserungsprogramms in einem Unternehmen ist es wichtig, folgende Aspekte zu beachten (vergleiche [Nati94, BaMc95]):

- **Motiviere das Programm an Verbesserungszielen**. Diese Verbesserungsziele beziehen sich häufig auf nicht direkt meßbare Größen, z.B. Kundenzufriedenheit. Sie können auf eine konkret meßbare Ebene hinunter verfeinert werden.
- **Fange klein an**. Die Beschränkung der Meß- und Verbesserungsaktivitäten zunächst auf sehr wenige Projekte ermöglicht die optimale Unterstützung dieser Projekte. Wenn diese ersten Projekte erfolgreich gelaufen sind, ist der Zeitpunkt für eine Verbreiterung gekommen.
- **Beziehe alle Mitarbeiter mit ein**. Dies baut die Motivation für ein solches Programm auf und ermöglicht gleichzeitig die Einbeziehung des Know-hows aller Mitarbeiter bei Verbesserungen.
- **Verbesserung der Produkte ist das Hauptziel.** Die Verbesserung der Prozesse ist Mittel zum Zweck. Somit darf Erfolg nicht nur an Prozeßverbesserungen gemessen werden, sondern muß sich auf Produktverbesserungen beziehen. Zu Beginn muß der Ist-Zustand der Produkte erfaßt werden, erst in zweiter Linie ist der Ist-Zustand der Prozesse wichtig.
- **Sichere dem Programm die Unterstützung des Managements**. Das Verbesserungsprogramm muß vom Management mitgetragen werden. Nur so können die für die Einführung notwendigen Ressourcen bereitgestellt werden und Verbesserungen dauerhaft eingeführt werden.
- **Erziele und verbreite frühe Erfolge**. Die ersten Resultate des Meßprogramms müssen verkündet werden, z.B. besseres Verständnis für die typischerweise auftretenden

Fehler oder für die Aufwandsverteilung Codieren / Testen. Allerdings muß auch klargesteilt sein, daß sofortige Verbesserungen allein durch die Einführung eines Meßprogramms nicht zu erwarten sind. Meßbare Verbesserungen werden erst später auftreten.

- **Ziel des Qualitätsverbesserungsprogramms ist die eigene Verbesserung und nicht der Vergleich mit Anderen.** Der Vergleich mit anderen Bereichen, auch unternehmensintern, macht nur auf einer sehr allgemeinen Ebene Sinn. Die Qualitätsmerkmale, die betrachtet werden, sind bereichs- und unternehmensspezifisch definiert.
- **Kündige künftige Änderungen an.** Ein Qualitätsverbesserungsprogramm zieht natürlich Änderungen mit sich. Diese Änderungen können auf Widerstände stoßen. Deswegen ist es wichtig, die zu erwartenden Änderungen rechtzeitig anzukündigen, damit das Management und die am Programm beteiligten Projektmitglieder sich damit auseinandersetzen können.
- **Richte rechtzeitig eine „Experience Factory" ein.** Die Einführung eines Meßprogramms ist ein erster Schritt zur kontinuierlichen Qualitätsverbesserung, aber die Einrichtung einer Organisationsform zur Sicherung von Software-Entwicklungs-Know-how darf nicht vernachlässigt werden. Dies ist wichtig, damit die beim Messen und Analysieren gewonnen Erfahrungen auch in einer Form festgehalten werden, die sie für spätere Projekte wiederverwendbar machen.
- **Weite das Programm langsam aus.** Verbesserungen benötigen Zeit. Die Durchführung von Meßprogrammen und das Sichern von Erfahrungen benötigt Ressourcen. Die Aktivitäten müssen den verfügbaren Ressourcen angepaßt sein.

5 Beispiele

Dieses Kapitel stellt drei Organisationen vor, die das QIP anwenden. Das QIP dient ihnen als Grundlage für angewandte Software-Engineering-Forschung und für industrielle Software-Qualitätsverbesserung. Die hier beschrieben Organisationen sind das Software Engineering Laboratory (SEL) der NASA in Greenbelt, USA, das Fraunhofer-Institut für Experimentelles Software Engineering (IESE) in Kaiserslautern, Deutschland, und das International Software Engineering Research Network (ISERN).

5.1 SEL

Das Software Engineering Laboratory (SEL) wurde 1976 von Goddard Space Flight Center der NASA (NASA/GSFC), Computer Sciences Corporation (CSC) und der University of Maryland (UMD) gegründet. Ziel des SEL ist es seit der Gründung, die Entwicklung von Software zur Steuerung von Satelliten meßbar zu verbessern. Das SEL - unter der Leitung von Frank McGarry (NASA/GSFC), Gerry Page (CSC) und Victor Basili (UMD) - gilt als

Initiator und erfolgreichste Einrichtung einer industriellen „Experience Factory". Dies wurde 1994 mit der Verleihung des 1. IEEE Process Achievement Awards gewürdigt. Das SEL hat signifikante Verbesserung der Software-Entwicklung (bzgl. Qualität, Produktivität, Produktivität etc.) erreicht und Methoden zur Qualitätsverbesserung wie QIP, GQM und Experience Factory formuliert. Viele weitere Organisationen (z.B. Motorola, Hughes, Robert Bosch, Allianz, Daimler-Benz) haben diese methodischen Ansätze von SEL übernommen und verwenden sie heute. Für detailliertere Informationen über das SEL und seine Arbeitsweise verweisen wir auf [BaMc95], ein Handbuch zum Messen und Bewerten [Nati94] sowie viele weitere Veröffentlichungen. Außerdem steht die WWW home page des SEL unter der Adresse http://groucho.gsfc.nasa.gov/Code_550/SEL_hp.html zur Verfügung.

5.2 FhE IESE

Das Fraunhofer Institut für Experimentelles Software Engineering (IESE) wurde 1996 in Kaiserslautern gegründet. Ziel des Instituts ist es, aufbauend auf den Erfahrungen des SEL, europäischen Firmen beim Aufbau industrieller Qualitätsverbesserungsprogramme zu helfen. Das IESE soll unter der Leitung von Prof. Dieter Rombach, der zuvor acht Jahre an maßgeblicher Stelle im SEL mitgearbeitet hat, zum Kompetenzzentrum für angewandte Software Engineering Forschung und Technologie-Transfer werden. Erste erfolgreiche Anwendungen der SEL-Methoden QIP, GQM und Experience Factory wurden bereits seit 1993 im Rahmen der Software-Technologie-Transfer-Initiative (STTI-KL) durchgeführt. Das enorme Interesse der Industrie an diesen Ansätzen hat 1996 zur Überführung der STTI-KL in ein selbständiges Fraunhofer-Institut geführt. Kernkompetenzen des IESE liegen in den Bereichen Software-Qualitäts- und -Prozeß-Engineering, Software-Entwicklungsumgebungen sowie Cleanroom Software Engineering, Requirements Engineering, domänenspezifischen Ansätzen und Reengineering/ Maintenance. Beispiele industrieller Kooperationsprojekte sind die Projekte mit Allianz Leben, Robert Bosch, Daimler-Benz, Ericsson, Siemens, Markant Südwest, Q-Labs und Tech-Math.
Für detailliertere Informationen verweisen wir auf Veröffentlichungen der STTI-KL und des IESE (Kontakt: Carsten Tautz: tautz@iese.fhg.de) sowie auf die WWW home page des IESE unter der Adresse http://www.iese.fhg.de.

5.3 ISERN

Das International Software Engineering Research Network (ISERN) wurde 1993 von mehreren universitären und industriellen Organisationen zur Förderung der Forschung im Bereich des Experimentellen Software Engineering gegründet. Die Mitglieder kooperieren in vielfältiger Form. Die Kooperationen reichen vom Austausch von Informationen, Werkzeugen und Mitarbeitern bis hin zu verteilten Experimenten und gemeinschaftlichen Forschungsprojekten. Verteilte Experimente, d.h. Replikationen bzw. Variationen von Ex-

perimenten, haben sich als notwendige Voraussetzung für die Weiterentwicklung des Experimentellen Software Engineering als Disziplin erwiesen. Viele Fragestellungen, wie z.B. die Bewertung der Effektivität bestimmter Software Engineering Ansätze (z.B. Cleanroom) unter verschiedenen Randbedingungen sind derartig komplex, daß einzelne Arbeitsgruppen damit überfordert sind. Verteilte Experimente haben bereits zu vorher undenkbaren Fortschritten geführt. Als Beispiel seien hier die Experimente zum Thema „Effektivität systematischer Software-Inspektionen" erwähnt, die gemeinsam von der University of Maryland (Victor Basili, Adam Porter) AT&T (Larry Votta), der University of New South Wales in Sydney (Ross Jeffrey) und der Univrsität Kaiserslautern (Dieter Rombach, Oliver Laitenberger) durchgeführt worden sind. Aufgrund dieser experimentellen Ergebnisse werden bereits bestimmte systematische Inspektionstechniken im Rahmen des IESE in die industrielle Praxis eingeführt. Interessenten können per email (isern@informatik.uni-kl.de) Informationen über ISERN erhalten.

6 Zusammenfassung

In diesem Beitrag wird ein Ansatz zur kontinuierlichen Qualitätsverbesserung von Software beschrieben, der Experimentelle Software Engineering Ansatz auf der Grundlage des Quality Improvement Paradigm. Das QIP wird im Vergleich mit anderen Ansätzen, die ähnliche Ansprüche bzgl. Qualitätsverbesserung stellen, bewertet. Die Entwicklung von Software mit hoher Qualität ist sehr komplex. Das QIP eignet sich besser als andere Ansätze zur Beherrschung dieser Komplexität, da es verschiedene wissenschaftliche Vorgehensweisen, die zum Aufbau von Software-Entwicklungs-Know-how erforderlich sind, in sich vereinigt. Das QIP benötigt zur Bewältigung der Aufgabe der kontinuierlichen Verbesserung drei Infrastrukturtechnologien: zielorientiertes Messen, explizites Modellieren und umfassende Wiederverwendung. Mit Hilfe dieser drei Technologien werden einmal erreichte Verbesserungen wiederholbar und nachweisbar.
Für die Verbesserung von Software-Entwicklungs-Know-how eignet sich der GQM-Ansatz zum Messen im Vergleich zu anderen zielorientierten Meßansätzen am besten. Die Prinzipien des GQM-Ansatzes führen zu einer zielgerichteten Erfassung der im geltenden Kontext relevanten Daten. Meßprogramme nach der GQM-Methode als Bestandteil von Qualitätsverbesserungsprogrammen nach QIP wurden in der Praxis erfolgreich durchgeführt, über Erfahrungen wird berichtet. Das QIP als grundlegender Ansatz zur kontinuierlichen Verbesserung wird von verschiedenen Organisationen mit Erfolg angewendet.

Bedanken möchten wir uns bei allen Mitarbeiterinnen und Mitarbeitern der Arbeitsgruppe Software Engineering. Vor allem bedanken wir uns bei Andreas Birk, der wertvolle Hinweise zur Verbesserung des Beitrags lieferte, bei Alfred Bröckers, Christiane Gresse, Martin Verlage und Isabella Wieczorek für ihre hilfreichen Kommentare sowie bei Dorothea Kilgore

für ihre wertvolle Unterstützung bei der Erstellung des Manuskriptes. Weiter bedanken wir uns bei den Mitgliedern des International Software Engineering Research Networks (ISERN), allen voran bei Victor Basili, der den in diesem Beitrag beschriebenen Ansatz begründete.

Referenzen

[AMI92] AMI Consortium. AMI; Metric Users' Handbook. AMI Consortium,1992.

[Basi85] Victor R. Basili. Quantitative evaluation of software methodology. In Proceedings of the 1st Pan-Pacific Computer Conference, Melbourne, Australien, September 1985.

[Bas92] Victor R. Basili. Software modeling and measurement: The Goal/Question/Metric paradigm. Technical Report CS-TR-2956, Department of Computer Science, University of Maryland, College Park, MD 20742, September 1992.

[Basi95] Victor R. Basili. The Experience Factory and its relationship to other quality approaches. In Marvin Zelkovitz, Herausgeber, Advances in Computers, vol. 41, Seiten 65-82. Academic Press,1995.

[BoBL76] B.W. Boehm, J. R. Brown und M. Lipow. Quantitative evaluation of software quality. In Proceedings of the 2nd International Conference on Software Engineering. IEEE Computer Society Press, Oktober 1976.

[BaCR94a] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. Experience Factory. In John J. Marciniak, Herausgeber, Encyclopedia of Software Engineering, Band 1, Seiten 469-467. John Wiley & Sons,1994.

[BaCR94b] Victor R. Basili, Gianluigi Caldiera und H. Dieter Rombach. Goal Question Metric Paradigm. In John J. Marciniak, Herausgeber, Encyclopedia of Software Engineering, Band1, Seiten 528-532. John Wiley & Sons,1994.

[BiKe95a] Andreas Birk und Ralf Kempkens. Introduction to Goal-Oriented Measurement. Tutorial Package. ESPRIT Project #9090 "PERFECT" und Universität Kaiserslautern, Kaiserslautern, Deutschland, 1995.

[BiKe95b] Andreas Birk und Ralf Kempkens. Participation of Project Teams in Measurement Programs. Tutorial Package. ESPRIT Project #9090 "PERFECT" und Universität Kaiserslautern, Kaiserslautern, Deutschland, 1995.

[BrDT96] Alfred Bröckers, Christiane Differding und Günter Threin. The Role of Software Process Modeling in Planning Industrial Measurement Programs. Proceedings of the Third International Software Metrics Symposium, Berlin, März 1996.

[BrLRV95] Alfred Bröckers, Christopher M. Lott, H. Dieter Rombach und Martin Verlage. MVP-L Language report version 2. Technischer Bericht 265/95, Fachbereich Informatik, Universität Kaiserslautern, 67653 Kaiserslautern, Deutschland, 1995.

[BaMc95] Victor R. Basili und Frank McGarry. The Experience Factory: How to build and run one, April 1995. Tutorial gehalten bei der 17. International Conference on Software Engineering (ICSE17), Seattle, Washington, USA.

[BaRo88] Victor R. Basili und H. Dieter Rombach. The TAME Project: Towards improvement-oriented software environments. IEEE Transactions on Software Engineering, SE-14(6):758-773, Juni 1988.

[BaWe84] Victor R. Basili und David M. Weiss. A methodology for collecting valid software engineering data. IEEE Transactions on Software Engineering, SE-10(6):728-738, November 1984.

[GrHW95] Christiane Gresse, Barbara Hoisl und Jürgen Wüst. A Process Model for GQM-Based Measurement. Technischer Bericht STTI-95-04-E, Software-Technologie-Transfer-Initiative, 67653 Kaiserslautern, Deutschland, Juli 1995.

[DiHL95] Christiane Differding, Barbara Hoisl und Chistopher M. Lott. Technology package „GQM„. Technischer Bericht, Software-Technologie-Transfer-Initiative, 67653 Kaiserslautern, Deutschland, Juli 1995.

[Dif93] Christiane Differding. Ein Objektmodell zur Unterstützung des GQM-Paradigmas. Diplomarbeit, Fachbereich Informatik, Universität Kaiserslautern, 67653 Kaiserslautern, Juni 1993.

[KoAk83] M. Kogure und Y. Akao. Quality Function Deployment and CWQC in Japan. Quality Progress, October 1983.

[McRW77] J. A. McCall, P. K. Richards und G. F. Walters. Factors in Software Quality. Technical Report RADC-TR-77-369, Rome Laboratory, Griffis AFB, NY, 13441, 1977.

[Muri80] Gerald E. Murine. Applying Software Quality Metrics in the requairements phase of a distributive system. Technical report, Metriqs Incorporated, 390 Oak Avenue, Carlsbad, California 92008, 1980.

[Nati95] National Aeronautic and Space Administration. Software measurement guidebook. Technical Report SEL-84-101, NASA Goddard Space Flight Center, Greenbelt MD 20771, Juli 1994.

[PCCW93] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, und Charles V. Weber. Capability maturity model, version 1.1. IEEE Software, 10(4):18-27, July 1993.

[RHMP85] Ronald A. Radice, John T. Harding, Paul E. Munnis und Richard W. Phillips. A programming process study. IBM Systems Journal, 24(2):91-101,1985.

[Romb91] H. Dieter Rombach. Practical benefits of goal-oriented measurement. In N. Fenton und B. Littlewood, Herausgeber, Software Reliability and Metrics, Seiten 217-235. Elsevier Applied Science, London, 1991.

[VanM95] Marco van Maris. GQM-DIVA, Ein Werkzeug zur Defintion, Interpretation und Validation von GQM-Plänen. Diplomarbeit, Fachbereich Informatik, Universität Kaiserslautern, 67653 Kaiserslautern, Deutschland, 1995.