5 Qualitätssicherung

Dieses Kapitel befasst sich mit der Qualitätssicherung im Allgemeinen, den Zusammenhängen zum Qualitätsmanagement und wichtigen Methoden der Umsetzung der gestellten Aufgaben.

Unter dem Begriff *Qualitätssicherung* versteht man gemäss DIN 55350 die "*Gesamtheit der Tätigkeiten des Qualitätsmanagements, der Qualitätsplanung, der Qualitätslenkung und die Qualitätsprüfung*". Die Umsetzung der Anforderungen, der Qualitätsvisionen und Ziele erfordert Werkzeuge, Methoden und Ressourcen, die durch die Qualitätssicherung bereitgestellt werden und sowohl die Produktqualität als auch die Prozessqualität betreffen. Während sich die Produktqualität primär auf (Teil-) Produkte bezieht, indem gleichförmige Produkte, d.h. Produkte von gleich bleibend hoher Qualität, hergestellt werden sollen, wendet sich die Prozessqualität an die Abläufe, um diese Produkte herstellen zu können.

Das Kapitel soll Projekt- und Qualitätsmanagern einen Überblick über die Qualitätssicherung und wichtige Werkzeuge vermitteln und das Verständnis für "Qualität" im Projektumfeld erhöhen.

Qualitätsfaktoren und deren Beurteilung. Je nach Rolle und Projektbeteiligung existieren zahlreiche Qualitätsforderungen an das Produkt und an das Projekt, die im gesamten Projektverlauf aber auch bereits bei der Definition von Anforderungen berücksichtigt werden müssen. Diese Qualitätsforderungen sind projektabhängig und variieren in deren Bedeutung und Wichtigkeit. Exemplarisch werden Qualitätsfaktoren als Basis für eine *Trade-Off* Abschätzung aus Anwendersicht ausgeführt.

Ablaufmodell Qualitätssicherung. Ein Ablaufmodell für die Qualitätssicherung stellt die Zusammenhänge zwischen der Softwareentwicklung bzw. den jeweiligen Phasen im Softwareentwicklungsprozess und der Qualitätssicherung dar. Aus den Qualitätsfaktoren wird exemplarisch ein entsprechendes Ablaufmodell im Hinblick auf die Produkt- und Prozessqualität erarbeitet.

Methode Reviews und Inspektionen. Die Umsetzung der Forderungen aus dem Ablaufmodell bzw. aus den Vorgaben der Qualitätsplanung erfolgt durch verschiedene Werkzeuge, wie beispielsweise Reviews und Inspektionen. Dieses Kapitel beschreibt grundsätzliche Typen und Vorgehensweisen bei der Durchführung von Reviews und Inspektionen, die bereits in frühen Phasen der Softwareentwicklung eingesetzt werden können um Fehler in Softwaredokumenten zu finden.

Techniken zur Anwendung von Reviews. Ausgehend von der allgemeinen Vorgehensweise bei der Durchführung von Reviews werden verschiedene Schritte beim Fehlerfindungsprozess herausgearbeitet und geeignete Hilfsmittel, wie beispielsweise Lesetechniken, vorgestellt. Weiterführende Informationen zu den Lesetechniken werden in Kapitel 12 beschrieben.

Methode Testen. Um Fehler im fertigen bzw. halbfertigen Produkt, dem Programm, zu finden, hat sich die Methode Testen bewährt. Dieser Abschnitt beschreibt Grundlagen zum Testprozess im Zusammenhang mit der Qualitätssicherung und stellt einen Rahmen für spezielle, weiterführende Themenbereiche vor. Diese Detailinformationen werden in den darauf folgenden Kapiteln genauer vorgestellt.

5.1 Qualitätsfaktoren und deren Beurteilung

Dieser Abschnitt beschäftigt sich mit Qualitätsfaktoren und deren Beurteilung im Projektumfeld. Abhängig von den verschiedenen Sichtweisen der Projektbeteiligten (beispielsweise Projektmitglieder, Kunde, Endanwender) stellen die verschiedenen Rollen Anforderungen an das Produkt, an das Projekt und an dessen Qualität. Diese unterschiedlichen Sichtweisen müssen im Softwareentwicklungsprozess geeignet berücksichtigt werden, um die Bedürfnisse der Beteiligten zu befriedigen und um eine entsprechende Qualität erreichen zu können. Diesem Ansatz folgend, werden diese Anforderungen als *Qualitätsfaktoren* bezeichnet.

Unter Qualitätsfaktoren versteht man also Eigenschaften eines Produktes oder einer Dienstleistung, die erfüllt sein müssen, um eine hohe Qualität sicherzustellen und somit, unter Berücksichtigung des Kunden als oberste *Beurteilungsinstanz* der Produktqualität, die Akzeptanz des Produktes oder der Dienstleistung zu gewährleisten. Die Qualitätsfaktoren unterscheiden sich je nach Anwendungsgebiet, Produktart, Kundenkreis usw.

Abbildung 5.1 zeigt die wesentlichen, allgemeinen Eigenschaften eines (Software-) Endproduktes ohne Berücksichtigung der konkreten Funktionalität aus der Sicht einer Anwendergruppe. Schon die Auswahl der Qualitätsfaktoren, also der Produkteigenschaften, die für die Akzeptanz des Endprodukts relevant sind, legt die Prioritäten bei der Entwicklung des Produktes fest. Aus verschiedenen Perspektiven ergeben sich unterschiedliche Qualitätsfaktoren, die auf das Endprodukt wirken und somit umgesetzt werden müssen. Aus der Zusammenschau der Faktoren aller Rollen lässt sich ein Modell über die gemeinsam interessante Qualität bilden, das die Leitlinie für den Nutzen des Produktes bildet. Dadurch haben die Entwickler die Chance, zwischen nutzbringender Qualität und unnötigem "Vergolden" zu unterscheiden.

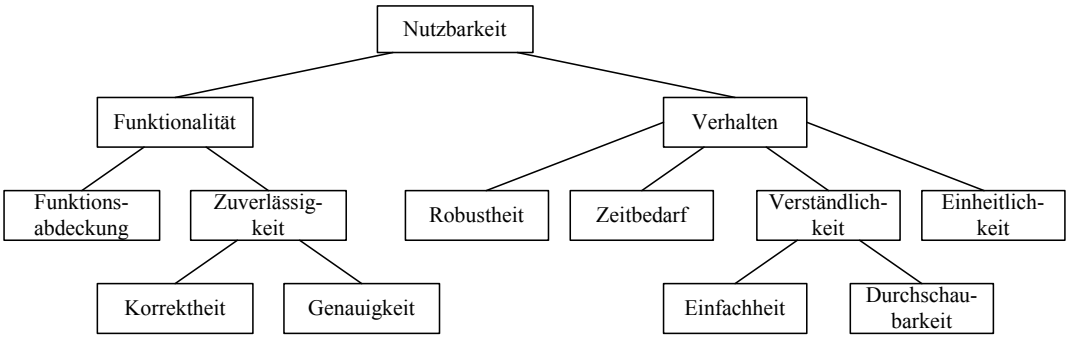

Abbildung 5.1: Relevante Eigenschaften aus Anwendersicht als Basis für eine Trade-Off Abschätzung

Erfüllt ein Softwareprodukt diese Eigenschaften, die intuitiv für den *Anwender* wichtig sind, assoziiert er *hohe Qualität* mit dem Produkt. Diese Eigenschaften sind jedoch nicht a priori gegeben sondern können nur durch den Einsatz geeigneter Methoden im Entwicklungsprozess "*hergestellt*" werden. Der Einsatz von Vorgehensmodellen, wie beispielsweise V- oder Spiralmodell, und der Einsatz von Qualitätsverbessernden Maßnahmen, wie beispielsweise Reviews und Testen, unterstützen das Entwicklungsteam dabei. An dieser Stelle muss jedoch noch

angemerkt werden, dass durch den Einsatz dieser Hilfsmittel *keine Qualität* erzeugt, sondern nur die Wahrscheinlichkeit zur Erstellung qualitativ hochwertiger Produkte und Dienstleistungen erhöht wird.

Wie bereits erwähnt ist die Relevanz von Qualitätsfaktoren von der Rolle bzw. Sichtweise der Rolle abhängig. Die IEEE beschreibt einen allgemeinen Ansatz zur Definition von Qualitätsfaktoren [IEEE, 1990]:

- *Korrektheit* (Correctness): Das Ausmaß, in dem Software ihre Spezifikation erfüllt
- *Zuverlässigkeit* (Reliability): Die Fähigkeit eines Systems, die verlangte Funktionalität unter gegebenen Randbedingungen für eine gegebene Zeit zu erfüllen
- *Effizienz* (Efficiency): Das Ausmaß, in dem ein System seine Leistungen mit einem Minimum an Ressourcenverbrauch erbringt
- *Integrität* (Integrity): Das Ausmaß, in dem ein System unberechtigte Zugriffe auf Programme und Daten bzw. deren unberechtigte Veränderung verhindert
- *Verwendbarkeit* (Usability): Die Leichtigkeit, mit der ein Benutzer die Bedienung eines Systems, die Vorbereitung von Daten dafür und die Interpretation seiner Ergebnisse lernen kann
- *Wartbarkeit* (Maintainability): Die Leichtigkeit, mit der ein System geändert werden kann, um Fehler zu beheben, seine Fähigkeiten zu erhöhen oder es an eine veränderte Umgebung anzupassen
- *Flexibilität* (Flexibility): Die Leichtigkeit, mit der ein System abgeändert werden kann, um es in Anwendungen oder Umgebungen zu benutzen, für die es nicht entwickelt worden ist
- *Testbarkeit* (Testability): Das Ausmaß, in dem ein System das Erstellen von Testbedingungen sowie die Durchführung von Tests zur Feststellung, ob die Bedingungen erfüllt sind, erleichtert
- *Portabilität* (Portability): Die Leichtigkeit, mit der ein System von einer Hard- bzw. Software-Umgebung in eine andere transferiert werden kann
- *Wiederverwendbarkeit* (Reusability): Das Ausmaß, in dem ein Stück Software in mehr als einem Programm oder Software-System verwendet werden kann
- *Verknüpfbarkeit* (Interoperability) :Die Leichtigkeit, mit der zwei oder mehrere Systeme Informationen austauschen und die ausgetauschten Informationen benutzen können.

Qualitätssicherungsmodelle versuchen nun, diese und weitere Qualitätsfaktoren zu unterteilen bzw. zusammenzufassen und zu klassifizieren. Diese Klassifikation bildet die Grundlage der Beurteilung der Qualitätseigenschaften bzw. Anforderungen an die Struktur von Software. Die Beurteilung dieser Qualitätseigenschaften erfordert jedoch messbare Größen und Metriken um einerseits quantifizierbare Aussagen treffen zu können und andererseits die Vergleichbarkeit zwischen Ergebnis und Ziel herzustellen.

Das *Goal-Question-Metrik-Modell* (GQM), das von V. Basili und D. Rombach entwickelt wurde, verfolgt das Ziel, ein System zur Messung von Software bzw. von Softwarequalität zu entwickeln.

Ausgehend von Qualitätszielen und geforderten Eigenschaften werden geeignete Fragen formuliert, deren Beantwortung als Nachweis für die Erfüllung dieser Eigenschaften verwendet werden kann. Unter Verwendung geeigneter Metriken wird eine Messgröße für den Erfüllungsgrad ermittelt. Beispielsweise kann für das Qualitätsziel "*Zuverlässigkeit*" die *Fehlerhäufigkeit* herangezogen werden. Als eine mögliche Metrik für den Nachweis der Fehlerhäufigkeit kann die Kenngröße *MTBF* (Mean time between failure, d.h. durchschnittliche Zeitspanne zwischen zwei auftretenden Fehlern) herangezogen werden [Hindel, 1996].

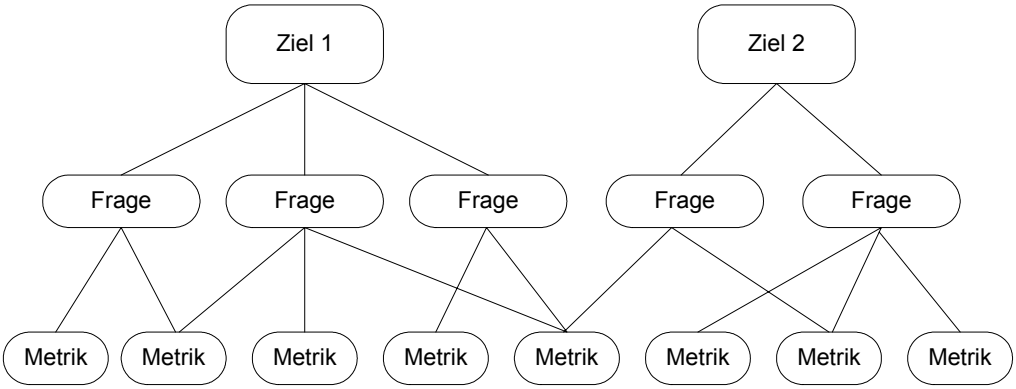

Abbildung 5.2: Goal Question Metric nach Basili[Basili et al, 1994]

Die Goal-Question-Metric (GQM) ist, wie in Abbildung 5.2 ersichtlich, hierarchisch aufgebaut. Das Modell besteht aus folgenden drei Stufen [Basili et al, 1994]:

- *Konzeptuelle Stufe (Ziele)*: Die Festlegung eines Ziels betrifft in der Regel Produkte, Prozesse oder Ressourcen und berücksichtigt dabei aktuelle Bedürfnisse und Sichtweisen. Die Ziele stellen dabei die Ausgangsbasis für den GQM Ansatz dar.
- *Operative Stufe (Frage)*: Die Fragestellungen charakterisieren das quantitativ zu erfassende Objekt (Produkt, Prozess, Ressourcen) entsprechend der zu untersuchenden Thematik aus der benötigten Sichtweise.
- *Quantitative Stufen (Metrik)*: In dieser Stufe werden konkrete gemessene Werte den Fragen der jeweiligen Ziele zugeordnet. Die Erfassung der Daten kann entweder subjektiv (beispielsweise unter Berücksichtigung einer bestimmten Sichtweise) oder objektiv erfolgen.

Ausgehend von dem allgemeinen Ansatz zur Ermittlung von Zielen, Fragen und Metriken existieren zahlreiche Modelle, so genannte Qualitätsmodelle, die bereits wesentliche Qualitätsfaktoren berücksichtigen. Abbildung 5.3 zeigt das Qualitätsmodell von McCall, das bereits 1977 in seiner ersten Version veröffentlicht wurde. McCall stellt den 3 Verwendungsarten (Produkteinsatz, Produktänderung und Produktwechsel) insgesamt 11 Qualitätsfaktoren gegenüber, die durch *Qualitätskriterien* beschrieben und messbar gemacht werden. Unter *Qualitätskriterien* verstehen wir also konkrete Eigenschaften oder Merkmale des Produktes, die zum Nachweis der Erfüllung der eher abstrakter formulierten Qualitätsfaktoren verwendet werden. Kennzahlen, die mit Hilfe von Metriken aus den Qualitätskriterien ermittelt werden, ermöglichen

auch eine direkte Vergleichbarkeit von verschiedenen Produkten oder Projekten und dienen somit als Grundlage für eine quantitative Abschätzung der Produktqualität.

Diese Qualitätskriterien werden also auf unterschiedliche Faktoren angewandt, da beispielsweise das Kriterium "Erweiterbarkeit" (*expandability*) sowohl für Produktänderungen als auch bei Produktwechsel berücksichtigt werden muss. Ausgehend von diesen Qualitätsfaktoren werden Metriken eingesetzt um eine quantitative Aussage über die Qualität treffen zu können. Beispielsweise kann die *Anzahl und Größe der Module* als Maßzahl für das *Kriterium Modularität* herangezogen werden. Die Modularität wirkt sich, laut McCall, auf die Qualitätsfaktoren *Testbarkeit und Flexibilität* bzw. *Wiederverwendbarkeit*, *Portierbarkeit* und *Kompatibilität* aus. Die Modularität spielt für den Einsatz des Produktes nur eine untergeordnete Rolle.

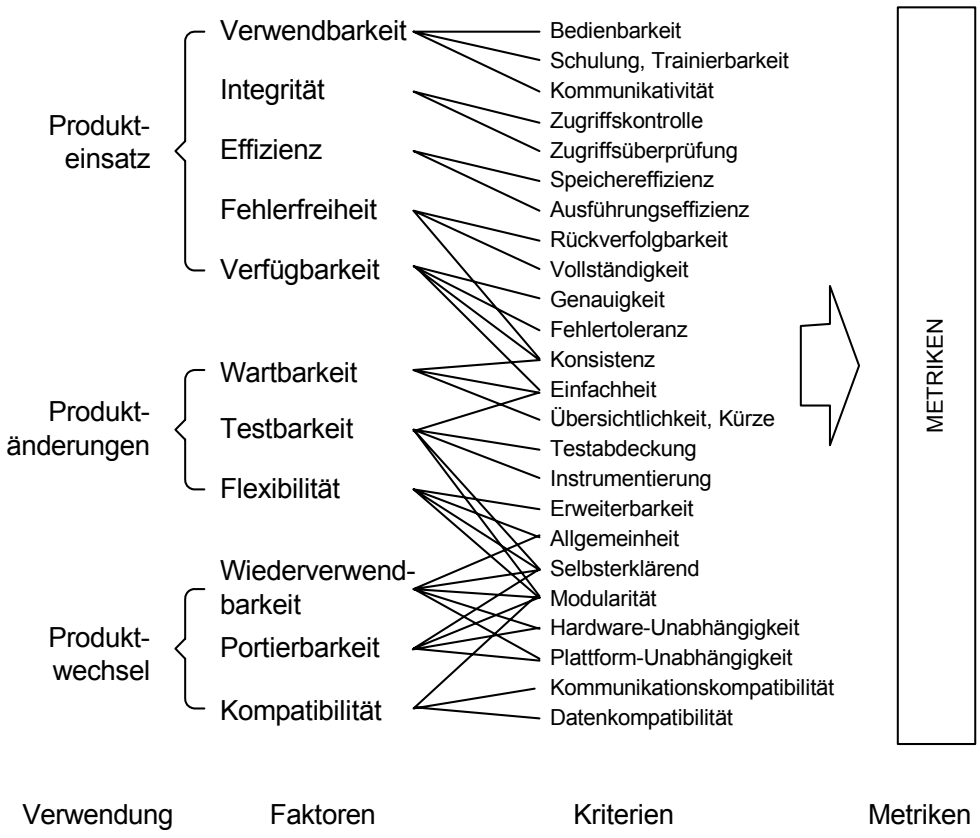

Abbildung 5.3: Qualitätsmodell nach McCall (1980) [Kitchenham et al, 2003] [Klaeren, 2003]

Die Definition von Qualitätskriterien und Qualitätsfaktoren ist allerdings abhängig vom Anwendungsgebiet des Softwareproduktes und muss im Rahmen der Projektplanung bereits berücksichtigt werden. Die wesentlichen Elemente des Modells nach McCall sind grundsätzlich in

allen Softwareprodukten zu finden; je nach Anwendungsbereich mit unterschiedlicher Gewichtung.

Quantitative Messgrößen eines Softwareproduktes sind im Allgemeinen ebenfalls notwendig, um eine ständige Verbesserung von Produkten und Prozessen, in diesem Fall von Entwicklungsprozessen, zu ermöglichen. Diese Forderung lässt sich auch aus den Qualitätsmanagementsystemen, wie beispielsweise CMM oder ISO 9000, ableiten, die Entscheidungen auf analytisch ermittelten Daten begründen und als integralen Bestandteil des Qualitätsmanagementsystems verstehen. Qualitätsmanagementsysteme werden ausführlich in Kapitel 13 behandelt.

Die Festlegung der Qualitätsfaktoren erfolgt im Regelfall während der Projektplanung. Die allgemeinen Zusammenhänge zwischen Qualitätssicherung und Softwareentwicklung werden meist durch ein Ablaufmodell der Qualitätssicherung visualisiert.

5.2 Ablaufmodell Qualitätssicherung

Um qualitativ hochwertige Software zu produzieren, ist es nicht ausreichend, die einzelnen – zum Teil bereits vorgestellten – Methoden und Modelle zu verwenden. Das Zusammenspiel dieser Ansätze ist notwendig, um den Softwareentwicklungsprozess als Einheit verwenden, analysieren und in weiterer Folge verbessern zu können.

Die Zusammenhänge zwischen Softwareentwicklung, Projektmanagement, Qualitätssicherung und Konfigurationsmanagement sind im beispielsweise "V-Modell des Bundes", das sich bereits als internationaler Entwicklungsstandard für IT-Systeme etabliert hat, beschrieben [IESE, 1997]. Dieses Modell umfasst neben Vorgehensmodellen ("*Was* ist zu tun?“) auch Methodenzuordnungen („*Wie* ist es zu tun?“) und funktionale Werkzeuganforderungen ("*Womit* ist etwas zu tun?"). Dieses dreistufige Standardisierungskonzept ermöglicht

- eine bessere Planbarkeit von IT-Projekten
- eine Reduktion bzw. Überschaubarkeit der Entwicklungskosten während des Produktlebenszyklus
- die Verbesserung der Qualität und
- die Verbesserung der Kommunikation zwischen Auftraggeber und Auftragnehmer.

Die Qualitätssicherung, die als Submodell einen integralen Bestandteil des V-Modells darstellt, definiert die Qualitätsfaktoren und -kriterien, die beispielsweise aus dem Qualitätsmodell nach McCall abgeleitet wurden, gibt Prüffälle und Prüfkriterien vor und überprüft die Produkte auf die Einhaltung der definierten Standards.

Softwarequalität wird durch *organisatorische* Maßnahmen ermöglicht, durch *konstruktive* Maßnahmen erreicht und durch *analytische* Maßnahmen ergänzt.

Organisatorische Maßnahmen umfassen sämtliche, durch das Management bereitgestellte, Rahmenbedingungen, die für ein effizientes Qualitätsmanagementsystem notwendig sind und somit die Basis für qualitätssichernde Maßnahmen darstellen. Im Regelfall werden Ressourcen und Mitarbeiter eingesetzt, die Richtlinien vorgeben, Schulungen und beratende Tätigkeiten

ausführen und bei qualitätsrelevanten Maßnahmen mitwirken und diese überwachen. Da Qualität nicht *in ein Produkt hineingeprüft* werden kann, ist es notwendig, durch *konstruktive Maßnahmen* die Erzeugung der Softwareprodukte zu fördern. Dadurch sollen ausserdem qualitätsrelevante Risiken und Qualitätsmängel vermieden werden.

Solche Qualitätsfördernde konstruktive Maßnahmen sind beispielsweise

- Vorgehens- und Phasenmodelle durch die Anwendung von Entwicklungsstandards
- Methoden und Werkzeugunterstützung in allen Phasen der Softwareentwicklung
- Richtlinien, Standards und Checklisten als organisatorische Hilfsmittel.

Da die Anwendung dieser konstruktiven Maßnahmen vom Projekt abhängt, ist es notwendig, diese bereits im Vorfeld zu definieren und im *Projektplan* zu dokumentieren.

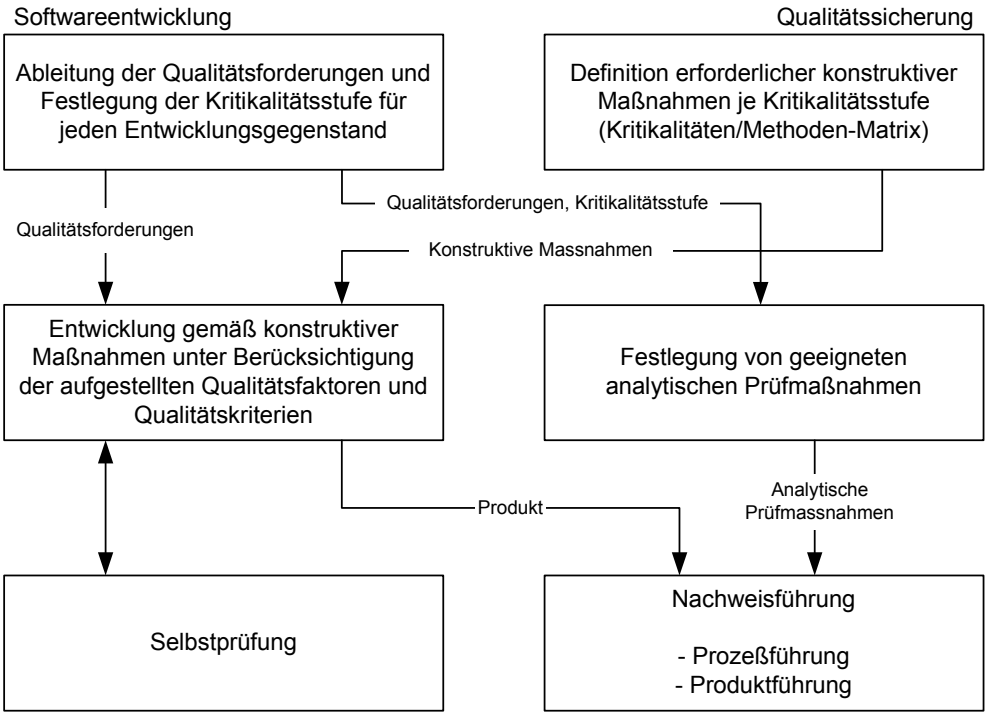

Abbildung 5.4: Analytische Methoden im V-Model [IESE, 1997]

Analytische Maßnahmen begleiten alle Phasen des Softwareentwicklungsprozesses und umfassen sowohl organisatorische als auch durchführende Tätigkeiten, um die Übereinstimmung des Produktes mit den Soll-Vorgaben zu überprüfen und nachzuweisen. Inspektionen, Reviews, Testmethoden usw. sind derartige analytische Maßnahmen, die auf die Prüfung, Bewertung und den (externen) Nachweis der Qualität der Prüfobjekte abzielen. Sie werden im Submodell QS (*Submodell Qualitätssicherung*) sowohl festgelegt als auch durchgeführt (siehe Abbildung 5.4).

Die analytischen Maßnahmen betreffen die Produkte und die Aktivitäten aller Submodelle. Die Qualitätssicherung (QS) soll und muss bereits in frühen Phasen der Softwareentwicklung eingesetzt werden und begleitet den Produktentwicklungsprozess durch alle Phasen. Die Ziele der Qualitätssicherung umfassen unter anderem

- frühzeitige Fehlererkennung und Fehlerbeseitigung,
- Einhaltung der vorgegebenen Qualitätseigenschaften, wobei es sich um inhaltliche oder um formale Eigenschaften handeln kann, die sich z.B. auf Wartbarkeit, Prüfbarkeit usw. beziehen,
- Modularität der Systementwicklung, da Qualität nachhaltig nur gewährleistet werden kann für solche Systeme, die nicht unnötig komplex sind.

Bei der Qualitätssicherung wird zwischen *Produktqualität* und *Prozessqualität* unterschieden. Entsprechend dieser Aufteilung werden Qualitätskriterien bestimmt, wobei es messbare, quantifizierbare (z.B. die Einhaltung gesetzlicher Bestimmungen) und nicht-quantifizierbare Kriterien (wie zum Beispiel Kundenzufriedenheit bei Beratungstätigkeiten) gibt.

> *Die Qualitätssicherung ist die Gesamtheit der Tätigkeiten des Qualitätsmanagements, der Qualitätsplanung, der Qualitätslenkung und die Qualitätsprüfung. (DIN 55350).*
>
> *Sie umfasst eine Vielzahl organisatorischer Maßnahmen, die bereits vor der Projektannahme mit den Planungsaktivitäten einsetzen und mit dem Abschluss des Projektes enden.*

Die Aufgaben der Qualitätssicherung lassen sich in drei Haupttätigkeiten einteilen: Planungs-, die Prüfungs- und die Lenkungsaktivitäten [IESE, 1997].

Durch *Planungsaktivitäten* werden allgemeine und produktspezifische Qualitätssicherungsmaßnahmen sowohl in zeitlicher Abfolge als auch in Art und Umfang festgelegt und ermöglichen somit erst entsprechende Prüfungen. Diese Planungaktivitäten finden im Vorfeld des Projekts bzw. in der Projektplanungsphase statt und werden entsprechend dem Projektfortschritt und den Ergebnissen der Prüfungen (siehe auch Lenkungsaktivitäten) nachjustiert.

Eine zweite wichtige Aktivität befasst sich mit der Durchführung der geplanten Prüfungen während des gesamten Projektverlaufs. *Prüfaktivitäten* dienen der Kontrolle der erstellten Produkte im Hinblick auf die Sollvorgaben und stellen die Grundlage für weitere Vorgehensweisen, beispielsweise weiterführende Phasen, dar. Prüfungen müssen in jedem Fall geplant werden, da sonst die Gefahr besteht, vor allem unter Zeitdruck, auf verschiedene Prüfungen zu verzichten, wodurch der Projektfortschritt nachhaltig negativ beeinflusst werden kann.

Abhängig von den Ergebnissen der Prüfungen, beispielsweise Freigabe- und Abnahmeprüfungen ist es notwendig, in den Projektverlauf einzugreifen um beispielsweise Prüfungen zu wiederholen oder Produkte, die eine unzureichende Qualität aufweisen, nachzubessern. Diese *Lenkungsaktivitäten* werden primär durch das Projektmanagement wahrgenommen und basieren auf Ergebnissen von Prüfaktivitäten. Beispielsweise können bei mangelnder Produktqualität, die bei Reviews festgestellt wurde, weiterführende Qualitätsmaßnahmen initiiert werden.

Ergebnisse von Prüfungen werden im Rahmen des Projektes sowohl intern als auch für externe Kunden im Rahmen der Nachweisführung verwendet um die Erfüllung der vorgegebenen Anforderungen objektiv nachweisbar zu machen und die Transparenz des Projektverlaufs zu gewährleisten.

Das Ablaufmodell zur Qualitätssicherung (Submodell QS) in Anlehnung an das *V-Modell* lässt sich, wie in Abbildung 5.5 dargestellt, visualisieren [IESE, 1997].

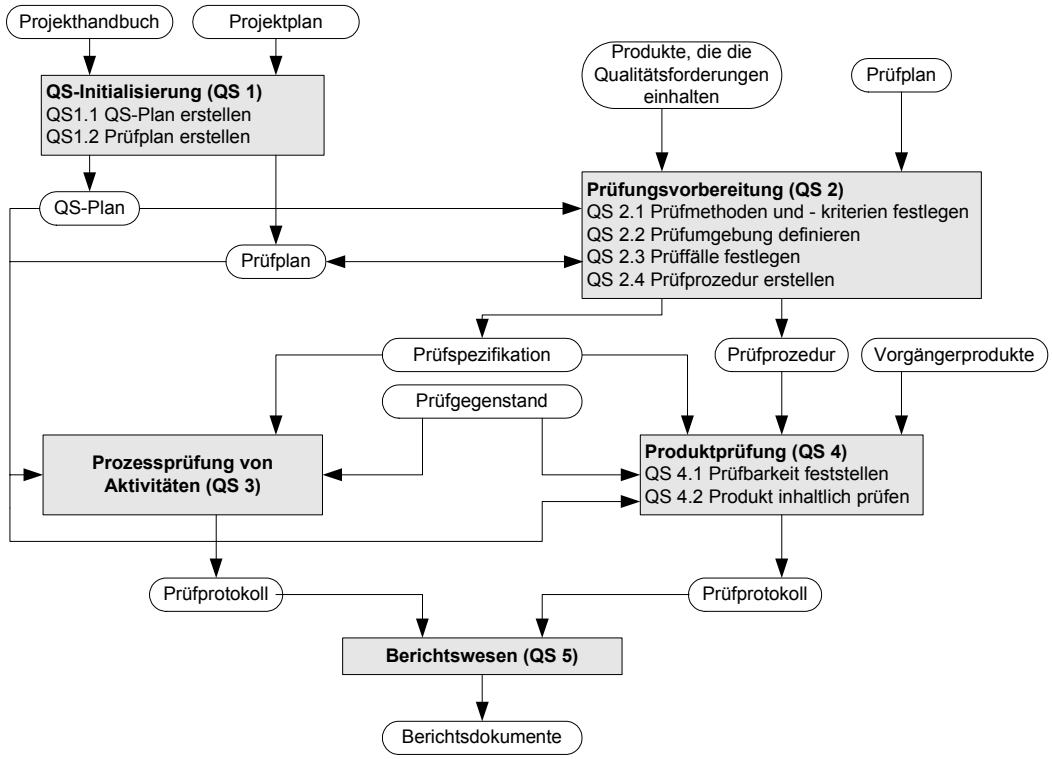

Abbildung 5.5: Submodell Qualitätssicherung gemäß V-Modell [IESE, 1997]

Das Submodell Qualitätssicherung (siehe Abbildung 5.5) umfasst die grundlegenden Stufen:

- *QS-Initialisierung (QS 1)*: Aufbauend auf dem Projektplan und dem Projekthandbuch, die das Projektmanagement jeweils bereitstellt, werden in dieser Stufe die organisatorischen und abwicklungstechnischen Rahmenbedingungen in Qualitätssicherungsplänen (QS-Plan) und Prüfplänen festgelegt.
- Zur *Prüfungsvorbereitung (QS 2)* gehören die Erstellung von Prüfspezifikationen und –prozeduren sowie die Vervollständigung des Prüfplans und der Prüfumgebung. Dabei sind nicht nur die entsprechenden Prüflinge vorzubereiten sondern auch Ressourcen, wie beispielsweise Termine, Dauer und Personen zu berücksichtigen.

- *Prozessprüfung von Aktivitäten (QS 3)*: Dabei wird festgestellt, ob vorgegebene Vorgehensweisen bei der Durchführung bestimmter Aktivitäten eingehalten werden.
- *Produktprüfung (QS 4):* Die Softwareprodukte sind entsprechend der Prüfspezifikation und –prozedur zu testen, das Ergebnis ist im Prüfprotokoll festzuhalten. Produktprüfungen beziehen sich nicht nur auf Software-Code sondern richten sich an alle Produkte, die im Rahmen der Softwareentwicklung erstellt werden, beispielsweise auch Anforderungsdokumente, Testpläne, Benutzerhandbücher, usw.
- *Berichtswesen (QS 5)*: In regelmäßigen Abständen sind die Prüfprotokolle, Zwischenberichte, usw., als Grundlage für die Lenkungsaktivitäten, auszuwerten und die Ergebnisse dem Projektmanagement vorzulegen. Die kontinuierliche Umsetzung dieser Maßnahmen gewährleistet einen transparenten und kontinuierlichen Projektfortschritt und die Möglichkeit, rechtzeitig auf auftretende Probleme zu reagieren und entsprechend korrigierend eingreifen zu können.

Hansen unterscheidet zu den im V-Modell angegebenen Schritten folgende zusätzliche Aktivitäten [Hansen, 1998]:

- *Durchführungsentscheidung*: Bestimmt, ob die nächste Aktivität durchgeführt werden kann. Dabei ist festzustellen, ob alle geplanten Produkte in der geforderten Form vorliegen, ob Kosten und Termine eingehalten und ob die nächsten Aktivitäten ordnungsgemäß geplant wurden.
- *Fertigprodukt prüfen*: Diese Prüfung soll die Erfüllung der Qualitätskriterien nachweisen. Auch hier gilt es, alle erzeugten und zur Prüfung vorgesehenen Softwareprodukte in der, während der Planungsphase vorgegebenen Weise, zu prüfen.

Die bisherigen Ausführungen zur Qualitätssicherung sind zwar im Grundsatz klar und folgen dem gesunden "Hausverstand", allerdings zeigt sich in der Praxis immer wieder, dass vor allem auf Prüfungen, auch wenn sie gut geplant sind, verzichtet wird, um einen meist recht eng gesetzten Zeitplan einzuhalten. Diese Vorgehensweise wirkt sich in der Regel negativ auf den gesamten Projektfortschritt aus, da sich Fehler, sofern sie unentdeckt bleiben, im weiteren Projekt verstärkt auswirken. Die Korrektur dieser Fehler resultiert – sofern sie, aufgrund der Schwere des Fehlers überhaupt noch korrigierbar sind – in der Regel in einer weiteren Verzögerung des Projektverlaufs. Es ist daher die Aufgabe des Projektleiters und des Qualitätssicherungsverantwortlichen (QSV) für die Umsetzung der Maßnahmen zu sorgen.

Der folgende Abschnitt stellt zwei wichtige Methoden für Fehlerfindung, Reviews und Inspektionen, vor. Diese beiden Methoden werden eingesetzt, um Fehler in allen Dokumentarten, also beispielsweise auch in Textdokumenten, in allen Phasen der Softwareentwicklung zu finden. Sie eigenen sich speziell auch für die Fehlerfindung in frühen Phasen der Softwareentwicklung, beispielsweise können damit Fehler bereits in Anforderungsdokumenten effizient gefunden werden.

5.3 Methode Reviews und Inspektionen

Dieser Abschnitt befasst sich mit zwei wichtigen Methoden der Softwarequalitätssicherung, den Reviews und Inspektionen, die bereits in frühen Phasen der Softwareentwicklung eingesetzt werden können, um Fehler schon bei der Erstellung des Softwaredokuments zu finden. Abhängig vom Fokus der Fehlererkennung werden diverse Ausprägungen und Rollen unterschieden.

In den Anfängen der Softwareentwicklung wurden Projekte, im speziellen Softwareprojekte, hauptsächlich *ad-hoc*, d.h. in enger Zusammenarbeit zwischen Auftraggeber und Entwickler, durchgeführt. Im Regelfall werden bei diesem Ansatz nur wenige Personen eingesetzt. Die Idee wird ohne die charakteristischen Merkmale eines Projektverlaufs direkt umgesetzt. Ein strukturiertes Vorgehen oder geeignete Dokumentation existiert im Regelfall nicht.

Mit der zunehmenden Verbreitung von Computern und Mikroprozessoren und steigender Komplexität der Produkte stiegen die quantitativen und qualitativen Anforderungen an Software, wobei die Prozesse um diese Software zu erzeugen immer noch dieselben waren. Dies führte meistens zu wenig befriedigenden Ergebnissen und zu Projektfehlschlägen, wie sehr hohe Kosten, zu lange Entwicklungsdauer und äusserst fehleranfällige Produkte. Aus diesem Grund entstand ein Bedarf an Kontrollmechanismen, um den Erfolg von Projekten zu gewährleisten. Diese Kontrollmechanismen werden heute mit den Begriffen Qualitätssicherung und Qualitätsmanagement zusammengefasst.

Reviews und Inspektionen sind Tätigkeiten, die innerhalb eines Projektteams angewandt werden, um Fehler zu finden. Das Ziel ist neben der eigentlichen Fehlerfindung der Zeitpunkt der Fehlerfindung. Je früher ein Problem erkannt und korrigiert wird, desto geringer sind die Kosten für dessen Korrektur. Beispielsweise kann ein schwerer Fehler (*critical defect*) im Anforderungsdokument, wenn er erst in der Integrationsphase erkannt wird, sehr hohe Kosten oder im ungünstigsten Fall das komplette Redesign des Produktes hervorrufen. Das primäre Ziel ist es also, diese Fehler möglichst frühzeitig zu erkennen und zu beheben.

Reviews und Inspektionen sind also Methoden der Qualitätssicherung, die während des gesamten Projektes zum Einsatz kommen. Sie sind nicht auf eine bestimmte Phase im Entwicklungsprozess beschränkt, werden aber bereits in der Analyse- und Designphase eingesetzt. Genauso wenig sind sie nur auf die Software-Entwicklung beschränkt, im Gegenteil, Reviews und Inspektionen haben sich bereits in nahezu allen technischen Prozessen bewährt.

5.3.1 Reviews

Dieser Abschnitt beschreibt wesentliche Aspekte von Reviews, gibt einen Überblick über wichtige Typen und Arten von Reviews und beschreibt, unter Berücksichtigung der verschiedenen Rollen, die Durchführung von Reviews.

Reviews sind qualitative Methoden und werden zur Verbesserung der Qualität sowohl von Prozessen als auch von Softwareprodukten eingesetzt. Im Mittelpunkt steht dabei die Überprüfung eines Softwareproduktes, eines Dokuments oder eines Programms, in – für Menschen – lesbarer Form gegenüber Vorgaben und Richtlinien, mit dem Ziel, Fehler zu finden. Die Review-Teilnehmer (Gutachter) verwenden dabei neben dem zu prüfenden Softwaredokument (*Prüfling*) auch unterstützende Unterlagen, wie Vorgabedokumente und Referenzunterlagen, Checklisten,

usw. Beispielsweise wird anhand der Entwurfsbeschreibung überprüft, ob im Lösungsansatz alle Anforderungsspezifikationen entsprechend den Richtlinien des Entwurfs umgesetzt worden sind.

Vor allem bei der Bewertung und Verbesserung von Entwicklungsprozessen kommt ihnen besondere Bedeutung zu, da sich Verbesserungen hier auf vielfältige Art und Weise auf das Endprodukt auswirken. Zudem ist die Weiterentwicklung von Prozessen auch eine Investition für die Zukunft, da auch nachfolgende Projekte davon profitieren.

Der Ablauf einer Review sollte wie der gesamte Entwicklungsprozess vorher festgelegt sein. Es ist für den Erfolg einer Review von entscheidender Bedeutung, dass sämtliche Vorbereitungen ordnungsgemäß durchgeführt wurden. Eine Review besteht also *nicht* nur aus einem Treffen freundlicher und kompetenter Menschen.

Arten von Reviews

IEEE definiert eine Review als "ein formelles Treffen, bei dem ein Produkt oder Dokument dem Benutzer, Kunden oder anderen interessierten Personen vorgelegt wird, um es zu kommentieren und abzusegnen" [IEEE, 610-1990]. Es kann eine Begutachtung des Managements und des technischen Fortschritts von Hardware-/Software-Entwicklungsprojekten sein.

Je nach Anwendungsbereich existieren zahlreiche verschiedene Ansätze für Reviews, die mit bzw. ohne Kunden stattfinden und in definierten Phasen zum Einsatz kommen. Abbildung 5.6 gibt einen Überblick über die wichtigsten Ansätze von Reviews.

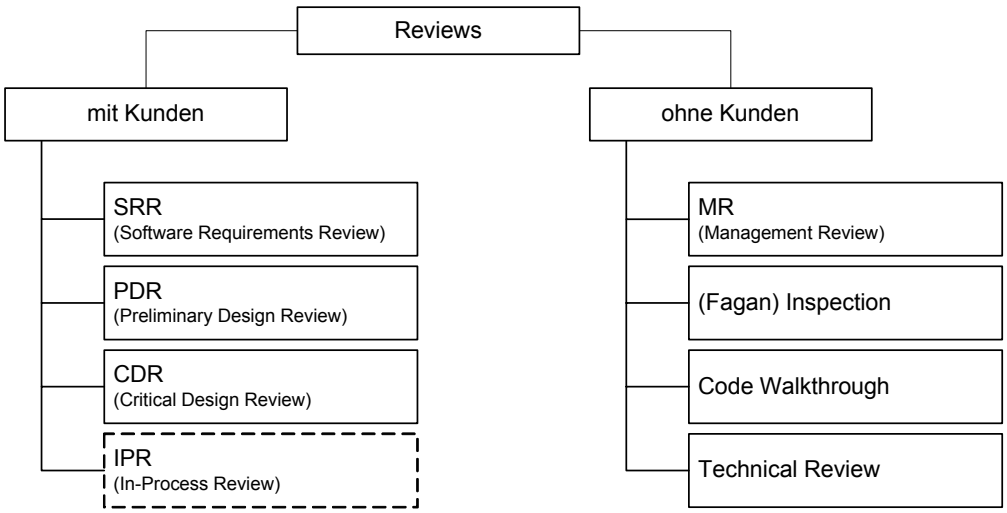

Abbildung 5.6: Arten von Reviews [Thaller, 2000]

Einerseits gibt es Reviews in Zusammenarbeit mit dem Kunden, andererseits auch verschiedene Arten von internen Reviews bei denen der Kunde nicht mit einbezogen wird. Die enge Zusammenarbeit mit dem Kunden in frühen Entwicklungsstadien des Software-Life-Cycle ist

ausgesprochen wichtig, damit das Ergebnis seinen Wünschen entspricht. Zuerst wenden wir uns den Reviewtypen zu, an denen der Kunde teilnimmt.

Sobald die Anforderungsanalyse abgeschlossen wurde, jedoch noch bevor sie als endgültige Version freigegeben wird, findet eine *Software Requirements Review (SRR)*, also ein Review mit dem speziellen Fokus auf die Anforderungsdokumente, statt. Üblicherweise wird die Überprüfung des Anforderungsdokuments vor der Entwurfsphase durchgeführt, um alle Unklarheiten sowohl seitens des Kunden, als auch seitens des Entwicklers zu diesem frühen Zeitpunkt auszuräumen. Nach dem SRR müssen die Ziele des Projektes allen beteiligten Personen bekannt sein.

Die nächste Review (*Preliminary Design Review (PDR)*) findet statt, sobald die Entwurfsphase beginnt, um den Entwurf der Softwarearchitektur und die Grobstruktur des Produkts zu überprüfen, bevor die Entwürfe im Detail ausgearbeitet werden. Dadurch wird sichergestellt, dass alle wesentlichen Anforderungen berücksichtigt und umgesetzt werden können.

Der Zeitpunkt nach der Entwurfsphase (bevor mit der Implementierung begonnen wird) stellt einen wichtigen *Meilenstein* dar und wird von einer „letzten“ Review, der *Critical Design Review (CDR)* begleitet. Dieser Review fällt besondere Bedeutung zu, weil es der letztmögliche Zeitpunkt ist, an dem der Kunde das Projekt stoppen kann, ohne unnötig viel Geld zu verschwenden. Je nach Größe und Komplexität des Softwareprodukts ist es möglich, PDR und CDR zusammenzufassen. In diesem Fall findet eine Review zum üblichen Zeitpunkt einer CDR statt. Der Sinn und Zweck einer CDR ist die Diskussion der detaillierten Spezifikation, weil danach die ausgesprochen zeit- und kostenintensive Phase der Implementierung des Programms beginnt.

Sehr große Projekte, deren Implementierung sich über einen besonders langen Zeitraum erstreckt, können so genannte *In-Process Reviews (IPR)* erforderlich machen, um Fortschritte in der Implementierung, Prototypen oder auch Testfälle dem Kunden zu präsentieren. Dies ist besonders wichtig um den Kunden über den Fortschritt des Projekts zu informieren.

Um zufrieden stellende Ergebnisse zu erzielen, müssen zusätzliche Methoden zur Verbesserung der Qualität und zur Überprüfung des Softwareprodukts hinsichtlich der Erfüllung der Anforderungen eingesetzt werden. Einige Mechanismen von internen Reviews (ohne Beteiligung des Kunden) sind Management-Reviews, technische Reviews, Inspektionen und Code Walkthroughs.

Management-Reviews dienen der formellen Bewertung des Projektplans oder auch um zu überprüfen, inwieweit der Projektstatus dem Projektplan folgt. Management-Reviews werden üblicherweise im Projektplan zu bestimmten Zeitpunkten des Software-Life-Cycle festgelegt, beispielsweise während der Analysephase, der Entwurfsphase usw. Je nach Bedarf können zusätzliche Management-Reviews festgesetzt werden, wenn ernsthafte Probleme auftreten oder unvorhersehbare Ereignisse dies erforderlich machen, z.B. wenn sich Verantwortlichkeiten verschieben oder wichtige Projektziele geändert werden.

Technische Reviews werden eingesetzt, um einen konkreten Bestandteil der Software zu bewerten, die Übereinstimmung von Spezifikation oder Standards mit dem Softwareprodukt selbst zu überprüfen und auch um Fehler zu finden.

Rollen bei der Durchführung von Reviews

Bei den Typen von Reviews wurden bisher nur die Rollen "Kunden" und sinngemäss "Entwickler" unterschieden. Es ist jedoch wichtig, dass innerhalb einer Reviewsitzung spezifische Rollen definiert werden, um einen effizienten und sinnvollen Ablauf der Review gewährleisten zu können. Die Rollen, die innerhalb einer Review eingenommen werden, basieren grundsätzlich auf den Rollen, die von Fagan bereits 1976 ermittelt wurden [Fagan, 1976], wurden jedoch weiterentwickelt bzw. modifiziert.

Die richtige Anzahl der Teilnehmer ist wichtig für die Effizienz einer Review. Zu viele Teilnehmer erhöhen zwar den Aufwand, bringen aber nur ein Minimum mehr an Nutzen bzw. an gefundenen Fehlern. IEEE schlägt eine Größe von 3-6 Personen vor [IEEE 1028-1997]. In der betrieblichen Praxis findet man meistens Größen von 4-5 Leuten. Allerdings gibt es hier Unterschiede, die durch die unterschiedlichen Review-Objekte verursacht werden. So sind zum Beispiel für Design-Reviews durchaus 5-6 Personen sinnvoll, weil Entwurfsdokumente und Designentwürfe schwieriger zu untersuchen sind als beispielsweise reiner Code. Abschließend kann man feststellen, dass Reviews mit 3-6 Personen, abhängig von der Art der Review und deren Zielsetzungen (Effizienz, Effektivität), durchgeführt werden sollten. Nun stellt sich noch die Frage welche Rollen die 3-6 Teilnehmer wahrnehmen müssen:

- *Moderator („keeper of process")*: Der Moderator ist für den Ablauf und die Organisation der Review zuständig. Er ist auch für Auswertungen und später für die korrekte Ausbesserung der Fehler zuständig. Der Moderator ist ebenfalls für die Auswahl der Teilnehmer sowie die Bereitstellung der benötigten Informationen und Unterlagen zuständig.

  Der Moderator sollte daher hauptsächlich organisatorische Fähigkeiten besitzen und auch als Diskussionsleiter fungieren können (daher: sicherstellen, daß die Review nicht vom Thema abdriftet; Konflikte lösen). Im Normalfall ist der Moderator nicht der Autor des zu untersuchenden Softwaredokuments.

- *Leser („keeper of focus and pace")*: Der Leser ist verantwortlich für die Aufbereitung der zu reviewenden Objekte für die Phase Meeting. Dies hat eine Beschleunigung des Ablaufes zum Ziel, bzw. auch eine Konzentration auf die wesentlichen Punkte. Weiters muss er dafür sorgen, dass die Geschwindigkeit des Reviews angemessen ist. Eine zu schnelle Review beinhaltet die Gefahr von zuviel übersehenen Fehlern, eine zu langsame Review ist ein Mehraufwand an Zeit ohne zusätzlichen Nutzen.

- *Schreiber („preserver of knowledge")*: Der Schreiber fungiert als Berichterstatter über den Verlauf der Review. Er muss alle Ergebnisse bzw. gefundenen Fehler mitprotokollieren und erstellt daraus ein Review Protokoll, das dann den Teilnehmern zur nochmaligen Überprüfung zur Verfügung gestellt wird. Dieses Dokument ist auch die Basis für die Nachbearbeitung der Review (Rework).

- *Gutachter („Reviewer")*: Der Gutachter hat generell die Aufgabe sich gut auf die Review vorzubereiten, und zwar anhand der vom Moderator weitergeleiteten Unterlagen. Weiters muss er sich natürlich aktiv an der Review beteiligen und auch bei der Erstellung des Review Dokumentes mitwirken.

- *Autor („author“)*: Der Autor ist der Urheber des zu reviewenden Objekts oder ein Repräsentant des Teams, das den Prüfling erstellt hat. Nimmt der Autor an dem Review teil, hat er die Aufgabe, auftretende Fragen zu klären und Hintergrundinformationen bereitzustellen, allerdings nicht aktiv an dem Review teilzunehmen oder gar Lösungen zu rechtfertigen oder dem Team zu "verkaufen".

Ein neuerer Ansatz von Laitenberger et al. gruppiert die Rollen beispielsweise genauer und nimmt den Autor des Review-Objektes zusätzlich dazu [Laiterberger et al, 1998]. In der Praxis wird aber eine Person mehrere dieser Rollen übernehmen müssen, daher kommt man, den Autor ausgenommen, zu den von Fagan vorgeschlagenen Rollen.

Durchführung von Reviews

Der Reviewprozess läuft in kontrolliertem Umfeld unter der Führung des Moderators ab. Unter Betrachtung des gesamten Prozesses sind folgende Schritte notwendig, um Reviews erfolgreich durchführen zu können:

- *Planung (planning phase)*: Wichtige Reviews werden bereits im QS-Plan bzw. im Projektplan festgelegt. Üblicherweise werden sie bei wichtigen Meilensteinen festgesetzt. Es liegt allerdings im Ermessen des Projektleiters, weitere Reviews durchzuführen (siehe auch Arten von Reviews). Bei der Planung müssen so genannte Arbeitpakete angefertigt werden. Diese Arbeitspakete beinhalten je nach Objekttyp den Prüfling, notwendige Referenzunterlagen bzw. Richtlinien und Vorgaben sowie angepasste Checklisten. Es ist die Aufgabe des Moderators, diese Arbeitspakete zusammenzufassen.
- *Initialisierung (initialisation phase)*: Für jede Review sollte es klar definierte Eingangskriterien geben, die ein Softwareprodukt erfüllen muss, bevor mit der Durchführung der Review begonnen werden darf. Solche Eingangskriterien können beispielsweise fordern, dass ein Code-Stück kompilierbar ist, bevor es inspiziert wird. Es ist die Aufgabe des Moderators dafür zu sorgen, dass der Prüfling, als Bestandteil des Arbeitspaketes, diese Eingangskriterien erfüllt.

  Nach Fertigstellung der Arbeitspakete wird das Reviewteam zusammengestellt, wobei alle wesentlichen Rollen besetzt werden müssen. Die Aufgabenteilung wird ebenfalls vom Moderator durchgeführt. Dabei muss jedoch beachtet werden, dass ein Aspekt jeweils von mindestens zwei Gutachtern betrachtet wird. Je nach Komplexität des Prüflings kann es notwendig sein, eine Einführungssitzung durchzuführen, bei der sowohl der Prüfling als auch die Unterlagen präsentiert werden. Das empfiehlt sich vor allem bei neuartigen oder komplexen Prüflingen.
- *Vorbereitung (preparation phase)*: Nach der Festlegung und Bekanntgabe der Daten für die Reviewsitzung (Arbeitspakete, Aufgaben- und Rollenverteilung, Ort, Termin, usw.) ist es die Aufgabe aller Reviewteilnehmer sich auf die Review vorzubereiten. In dieser Vorbereitungsphase untersuchen die Reviewer den Prüfling nach den ihnen zugeteilten Rollen und Aspekten und markieren gefunden Mängel. Dadurch wird die Effizienz der Review bei der eigentlichen Sitzung gesteigert.

  Beispiele für Mängel in Dokumenten (z.B. Anforderungsdokumente):

- Abweichungen von Richtlinien und Vorgabedokumenten
- Rechtschreibung
- Falsche / fehlende Begriffsdefinitionen, Referenzen, Verweise
- Inhaltliche Mängel

Beispiele für Mängel in Code-Dokumenten (z.B. Module, Schnittstellen):

- Abweichung von Richtlinien, Namenskonventionen, Kommentaren
- Strukturelle und inhaltliche Mängel

- *Reviewsitzung (review)*: Während der Reviewsitzung präsentiert der Leser das Softwaredokument (den Prüfling) und die entsprechenden Punkte auf der Checkliste; der Moderator leitet die Diskussion. Der Schreiber protokolliert die Erkenntnisse, und alle Reviewer beschließen gemeinsam, ob und welche Nachbearbeitung des Dokuments notwendig ist (Prüfling akzeptiert, Prüfling nach Überarbeitung akzeptiert oder Folgereview des Prüflings nach Überarbeitung).

  Eine derartige Reviewsitzung soll üblicherweise *nicht länger als zwei Stunden* dauern, da dann die Aufmerksamkeit der Teammitglieder stark abnimmt. Bei Bedarf können mehrere Reviewsitzungen durchgeführt werden.

- *"Die dritte Stunde" (meeting)*: Das Ziel einer Review ist die Fehlerfindung und nicht die Korrektur oder die Diskussion von Lösungsmöglichkeiten. Um die Möglichkeit der Diskussion zu schaffen, wird gegebenenfalls nach Beendigung der Reviewsitzung noch die "dritte Stunde" angehängt. Dieser Zeitraum ist für Diskussionen, Lösungsvorschläge und Feedback für den Reviewprozess gedacht. Dieses Meeting ist allerdings kein zwingender Bestandteil eines Reviewprozesses.

- *Review-Bericht (reporting)*: Nach der Reviewsitzung fertigt der Schreiber das Protokoll an und modifiziert es so lange, bis alle Reviewteilnehmer ihre Zustimmung geben. Nebenbei werden noch offene Punkte aus der Review-Diskussion geklärt und gegebenenfalls dem Protokoll hinzugefügt.

- *Nacharbeit (re-work)*: Der Autor des Dokuments nimmt die beschlossenen Änderungen oder sonstigen Nacharbeiten vor, der Moderator überprüft die Fertigstellung des Protokolls und die Durchführung der Nacharbeiten. Wenn das Dokument im Zuge der Nacharbeiten signifikant verändert wird, kann der Moderator eine weitere Reviewsitzung einberufen.

Sämtliche im Rahmen der Review erstellten Dokumente werden als Reviewpaket zusammengefasst und nach Abschluss der Review der Qualitätssicherung (QS) übergeben. Dort wird die ordnungsgemäße Durchführung der Review überprüft und eventuell für spätere Reviews der Reviewprozess verbessert.

5.3.2 Inspektionen

Inspektionen oder Software-Inspektionen weisen eine hohe Ähnlichkeit zu Reviews bzw. Technischen Reviews auf. Auch sie werden verwendet, um Fehler in frühen Stadien der

Softwareentwicklung zu finden und ein Softwareprodukt im Hinblick auf die Einhaltung von Standards und Vorgabedokumente (z.B. Anforderungsdokumente) zu überprüfen. Der Ablauf von Inspektionen ist formaler und erlaubt den direkten Vergleich zu Normen und Standards. Aufgrund des hohen Formalismus und relativ geringer Freiheitsgrade bei der Durchführung von Inspektionen, ist diese Methode auch für Schulungszwecke innerhalb des Projektteams gut einsetzbar, da sowohl der primäre Nutzen von Inspektionen – die Fehlerfindung – als auch das Kennenlernen des Produkts und der Methode unterstützt wird.

Inspektionen werden nach genau vordefinierten Schritten durchgeführt. Dabei sind die jeweiligen Phasen, als auch die zu ermittelnden Messdaten und die jeweiligen Rollen genau vordefiniert. Inspektionen weisen bezüglich Prozessablauf und der Teilnehmerzahl einen höheren Detaillierungsgrad auf. Diese Aufteilung kann jedoch ebenfalls für Reviews eingesetzt werden.

Arten von Inspektionen

Die *Fagan Inspection* ist die ursprüngliche Form von Inspektionen und bildet die Basis für die meisten anderen Inspektionstechniken. Bei Inspektionen werden dieselben Rollen, wie sie bereits bei im Kapitel „Reviews“ beschrieben wurden, eingesetzt (siehe Abschnitt 5.3.1).

Das *Active Design Review (ADR)* konzentriert sich, wie der Name schon sagt, hauptsächlich auf Inspektionen eines (Software-) Designs [Parnas et al, 1985]. Dieser Ansatz versucht, folgende Problemfelder der Fagan Inspektion zu beheben:

- Informationsüberladung der Inspektoren in der Vorbereitungsphase
- Aufgrund mangelhaften Zielbewusstseins sind Verbesserungen nur schwer umsetzbar
- Die soziale Interaktion dominiert bei grossen Gruppen auf Kosten des Inspektionsobjektes.

Um diese "Nachteile" zu verbessern wird der Inspektionsprozess neu strukturiert und in mehrere kleine Teilprozesse mit angepasstem Fokus auf die jeweiligen Softwaredokumente aufgeteilt. Jeder dieser Teilprozesse besteht, angelehnt an den Ablauf der von Fagan vorgeschlagen wurde, aus 3 Schritten. Anfänglich bekommen die Inspektoren eine grobe Einführung über den Teil, der gerade behandelt wird. Danach müssen die Inspektoren das Softwaredokument, mit Berücksichtigung des aktuellen Fokus, durchgehen. Dieser Schritt sollte anhand eines Leitfadens durchgeführt werden, um ein Abschweifen vom Fokus zu vermeiden. Im dritten Schritt werden die gefundenen Probleme in einem kleinen Meeting zwischen dem Designer und den Inspektoren erörtert.

Eine Reduktion der Inspektoren wird durch die *Two-Person Inspektion* umgesetzt. Dabei gibt es keinen Moderator, sondern ausschliesslich den Autor des Objektes und einen Inspektoren. Diese Methode ist effizient für kleine Unternehmen und kleine Softwareprojekte mit geringem Budget. Diese Methode wird auch als Vorstufe zu Inspektionsmethoden mit einer größeren Personenanzahl vorgeschlagen [Bisant et al, 1989].

Die *N-Fold Inspektion* zielt darauf ab, eine Anzahl n kleiner Inspektionsteams einzusetzen, die jeweils der Methode von Fagan folgen. Der Grundgedanke geht davon aus, dass kleine Gruppen höhere Fehlerfindungsraten aufweisen als vergleichsweise grosse Gruppen. Diese Methode wird von seinen Erfindern Martin und Tsai hauptsächlich für besonders kritische Projekte vorgeschlagen, nachdem sie relativ zeit- und kostenintensiv ist [Martin et al, 1990]. Da die n-fache

Anzahl von Personen (parallele Teams) für diese Inspektion benötigt wird, ist diese Methode nur sinnvoll, wenn sie auch den entsprechenden Nutzen bringt. Sie wird beispielsweise bei sicherheitsrelevanten Anwendungen oder bei Komponenten, die als Grundlage für den weiteren Projektfortschritt wesentlich sind, eingesetzt. Nicht gefundene Fehler in den Softwaredokumenten könnten möglicherweise gravierende Folgen nach sich ziehen (Projektverzögerungen bis zum Projektstopp oder Schäden an Personen und Maschinen usw.).

Eine Mischform aus den bisher genannten Inspektionsarten ist die so genannte *Phased Inspektion*, die von Knight und Myers entwickelt wurde [Knight et al, 1993]. Dabei wird das Softwaredokument in mehreren Schritten (so genannten *Phasen*) untersucht. Eine normale Inspektion kann bis zu 6 Phasen haben, wobei jede Phase ein bestimmtes Ziel verfolgt. Die Inspektoren untersuchen das Softwaredokument nur im Hinblick auf das jeweilige Ziel. Solange das aktuelle Ziel nicht erreicht ist und noch nicht alle Korrekturen erledigt sind, erfolgt keine Freigabe für die nächste Phase. Jede Phase wird weiters in zwei Teile aufgeteilt. Im ersten Schritt vergleichen die Inspektoren einzeln das Softwaredokument mit einer Checkliste auf Übereinstimmung. Danach vergleichen die Inspektoren das Objekt mit verschiedenen Checklisten, die anschließend in einem Meeting verglichen werden.

Ablauf von Inspektionen

Durch die Ähnlichkeit von Inspektionen und Reviews erfolgt auch die Vorgangsweise in ähnlicher Weise. Thaller beschreibt Fagans Inspektionsansatz in folgenden 6 Schritten [Thaller, 2000]:

- *Planungsphase (Planning Phase)*: Nach der Fertigstellung eines Dokumentes wird ein Inspektionsteam ernannt. Ein Teammitglied übernimmt die Rolle des Moderators und ist verantwortlich für den organisatorischen Ablauf.
- *Einführung (Tutorial)*: Neue Teammitglieder oder Teams die mit der Methode nicht vertraut sind, lernen die Grundregeln und Techniken der Inspektion. Diese optionale Einführung kann einerseits zum Verständnis der Methode und andererseits zum besseren Verständnis der Anwendungsdomäne oder des Inspektionsobjekts (Softwaredokument) eingesetzt werden.
- *Vorbereitungsphase (Preparation Phase)*: In einem Zeitraum von ungefähr zwei Stunden bereiten die Inspektoren die nötigen Hilfsmittel vor, die sie später für die Inspektion brauchen. Bei hoher Komplexität des Softwaredokuments oder bei unbekannter Anwendungsdomäne kann diese Vorbereitungsphase bei Bedarf ausgedehnt werden. Das primäre Ziel dieser Phase ist die Einarbeitung in den Anwendungsbereich und die Auseinandersetzung mit Inspektionen.
- *Ausführung (Operation)*: Unter der Leitung des Moderators inspiziert das Team das Softwareprodukt. Auch bei der Inspektion ist die Fehlerfindung das Hauptziel. Alle Fehler werden dokumentiert und dabei nach Typ, Schwere, Auswirkung, usw. katalogisiert. Dabei geht es nicht darum, mögliche Verbesserungen oder Korrekturen zu machen, sondern nur um die Erfassung der Fehler. Die subjektive Entscheidung über die Verwendbarkeit des Dokumentes, d.h. ob eine Überarbeitung notwendig ist oder das Dokument freigegeben werden kann, beendet die Ausführungsphase.
- *Überarbeitung (Rework)*: Basierend auf den notierten Mängeln überarbeitet der Autor das inspizierte Dokument.

- *Überprüfung der Richtigkeit (Verification)*: Nachdem alle Mängel behoben worden sind, überprüft der Moderator die Modifikationen und kann gegebenenfalls einen zweiten Inspektionszyklus (die Re-Inspektion) initiieren.

Die Technische Dimension von Software Inspektionen

Inspektionen sind *formale*, *effiziente* und *wirtschaftliche* Methoden um Fehler im Design und Code zu finden [IEEE, 1028-1997]. Inspektionen sind besonders dafür geeignet, Fehler in frühen Phasen der Softwareentwicklung, speziell in der Designphase aber auch in Software-Code, zu finden [Parnas et al, 1985]. Gilb et al. beschreiben Inspektionen als "Blicke durch ein Mikroskop" auf das Softwareprodukt mit dem Ziel, Fehler zu erkennen. Deshalb sehen Gilb et al. die Inspektionstechniken als eine Methode zur Fehlererkennung in allen Phasen des Software Life-Cycle [Gilb et al., 1993].

Laitenberger et al. beschreiben die technische Dimension der Software Inspektion in Abbildung 5.7 als einen Prozessrahmen für den *Inspektionsprozess*, der die *Produkte*, *Lesetechniken* und die *Rollen* innerhalb eines Inspektionsteams zeigt [Laitenberger et al., 1999; Laitenberger, 1998].

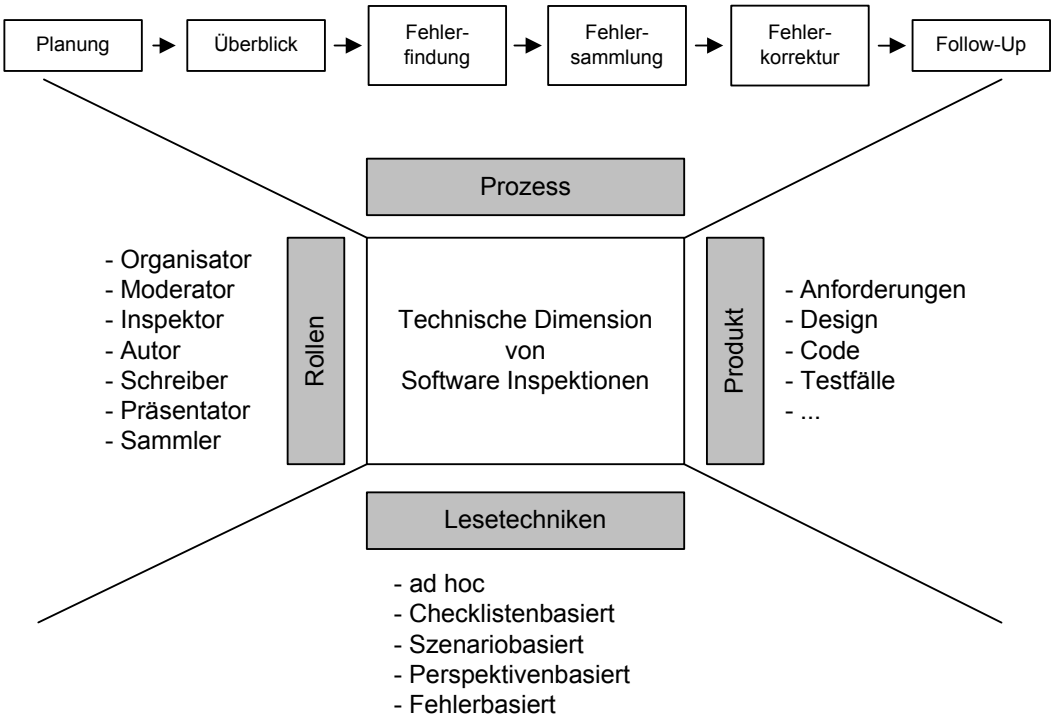

Abbildung 5.7: Technische Dimension von Software Inspektionen [Laitenberger et al., 1999]

Laitenberger et al. lehnen sich bei dem Inspektionsmodell an Fagans Inspektionsablauf an und berücksichtigen beim Inspektionsprozess die beteiligten Personen (Rollen), Produkte

(Softwaredokumente) und unterstützende Methoden (z.B. Lesetechniken) [Laitenberger, 1998], unterscheiden sich jedoch von Fagans Ansatz.

- Die *Planungsphase* zielt darauf ab, den eigentlichen Ablauf der Inspektion festzulegen und ist, wie bereits erwähnt, die Aufgabe des Moderators. Nachdem die Eingangsbedingungen erfüllt sind, muss das Inspektionsteam, unter Berücksichtigung der Fähigkeiten der Inspektoren (*inspector capabilities*), gebildet werden. Nach Fertigstellung der Inspektionsplanung werden die Arbeitspakete an die Gruppenmitglieder verteilt.

- Beim ersten Treffen ("kick-off-meeting") gibt der Autor einen ersten *Überblick* über die zu untersuchenden Softwaredokumente [Gilb et al, 1993]. Laut Laitenberger ist es sinnvoll mit dieser Phase fortzufahren wenn (i) das zu untersuchende Produkt sehr komplex und nur sehr schwer zu verstehen ist und (ii) es ein Teil eines großen Softwaresystems ist. In beiden Fällen kann der Autor auch die Zusammenhänge erklären [Laitenberger, 1998].

  Fagan unterstützt solche kick-off Meetings nicht, da sie eine unabhängige Inspektion verhindern könnten. Stattdessen unterstützt er "*Tutorials*" oder "*Vorbereitungsphasen*" für die Teammitglieder, damit diese mit dem Inspektionsprozess im Allgemeinen vertraut werden. [Fagan, 1976].

- Das Ziel der *Fehlerfindung* ist der eigentliche Kernpunkt der Softwareinspektion. Im Vergleich zu Fagan (Phase "Ausführung") wird diese Phase in eine Fehlerfindung und Fehlersammlung aufgeteilt.

  Unter "Fehlerfindung" versteht man die Tätigkeit, das zu inspizierende Softwaredokument zu untersuchen und im Hinblick auf Fehler zu bearbeiten. Der Fehlerfindungsprozess kann entweder individuell oder als Teamprozess stattfinden. Laut Fagan dominieren positive Effekte, wenn die Fehlerfindung als Teamprozess stattfinden, da das Team mehr Fehler finden kann als dieselbe Anzahl individueller Inspektoren. Empirische Untersuchungen, die von Votta (1993) durchgeführt wurden, ergaben jedoch keine signifikanten Unterschiede zwischen individuellem und Teamprozess [Biffl, 2001]. Der dritte Ansatz ist die Kombination beider Fehlerfindungsprozesse um die jeweiligen Vorteile zu nutzen. Durch individuelle Fehlersuche werden entsprechende Fehler gefunden, die anschliessend in einem Teammeeting besprochen werden und gruppendynamische Vorteile nutzen können.

- *Fehlersammlung*: Der zweite Schritt des Fehlerfindungsprozesses ist die Fehlersammlung, die je nach Inspektionsansatz individuell oder als Teammeeting durchgeführt werden kann. Durch Fehlersammlung werden nur *wirkliche* Fehler berücksichtigt und vermeintliche Fehler eliminiert. Diese Fehlersammlung erfolgt in vielen Fällen im Rahmen eines Teammeetings, in dem eine so genannte *Teamfehlerliste,* also eine Ansammlung abgestimmter, wirklicher Fehler im Dokument erstellt wird. Ein alternativer Ansatzpunkt sind so genannte *nominale Teamfehlerlisten*, die, unter Wegfall eines Teammeetings, alle individuellen Fehlerlisten zusammenfassen.

- *Fehlerkorrektur*: Basierend auf der (nominalen) Teamfehlerliste passt der Autor sein Softwaredokument an und legt es dem Moderator vor.

- *Follow-Up*: Das ist eine (optionale) Möglichkeit, die Veränderungen des Autors noch einmal von dem Inspektionsteam überprüfen zu lassen. Es wird festgestellt ob die Fehler von der Liste beseitigt wurden.

Inspektionen im Qualitätsmanagement

Inspektionen werden als Werkzeug zur Fehlerfindung eingesetzt und können somit zur Nachweisführung in der Qualitätssicherung bzw. im Qualitätsmanagement verwendet werden. Zusätzlich zu Fagans Basisprozess und Laitenbergers Unterdimensionen beschreibt Biffl die Integration von Inspektionen im Projekt- und Qualitätsmanagementumfeld [Biffl, 2001].

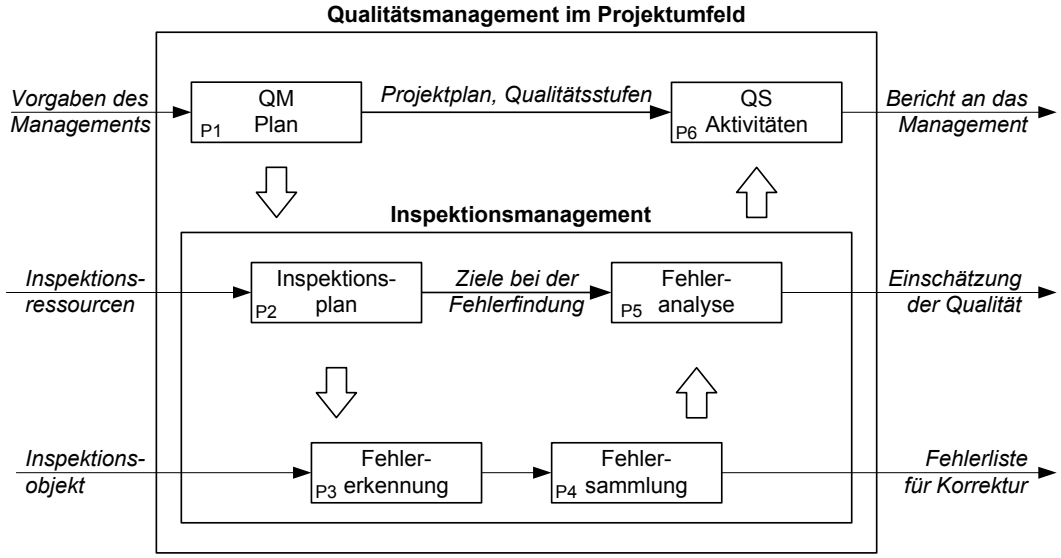

Abbildung 5.8: Inspektionsplanung und Kontrolle [Biffl, 2001]

Abbildung 5.8 beschreibt die zwei Managementansätze *Qualitätsmanagement* und *Inspektionsmanagement*: Durch das Qualitätsmanagement wird entschieden, ob und welche Maßnahmen zur Qualitätsverbesserung, in diesem Fall Inspektionen, in dem Projekt eingesetzt werden sollen. Das Inspektionsmanagement ist für die konkrete Umsetzung der Inspektion im Projektumfeld zuständig und verantwortlich.

Im dargestellten Modell treten drei verschiedene Schichten auf, die verschiedene Betrachtungsweisen hinsichtlich des Inspektionsprozesses und des Inspektionslevels repräsentieren.

1. Auf Projektmanagementebene werden grundlegende Vorgaben für Qualitätssichernde Maßnahmen definiert und Ergebnisse aus konkreten Prozessen bewertet. Diese Schicht betrachtet die Schnittstelle zwischen dem Risikomanagement eines Projekts und den Qualitätsmaßstäben der gesamten Firma. Die jeweiligen Vorgaben an das Projekt werden

einerseits im Projektplan und Qualitätsmanagementplan verankert (*P1*) und im Projektverlauf umgesetzt.

Ausgangsseitig werden Ergebnisse (*P6*) herangezogen, um die Qualität des Projektes, Prozesses oder Produktes zu bewerten und als Grundlage für Verbesserungen (z.B. in Form einer Wissensdatenbank) zu verwenden, aber auch um grundlegende Entscheidungen über den Einsatz von beispielsweise Inspektionen zu treffen (beispielsweise Kosten/Nutzen - Überlegungen, Prozessverbesserungen, Fehleranalyse, Entscheidung über weitere QS-Maßnahmen, usw.).

2. Das eigentliche Inspektionsmanagement ist für die Durchführung der Inspektion zuständig. Für eine erfolgreiche Umsetzung müssen alle notwendigen Rahmenbedingungen (Inspektionspersonal, Ressourcen, Unterlagen, usw.) bereitgestellt und ein angepasster Inspektionsplan (*P2*) erstellt werden. Dabei müssen neben den eigentlichen Prüflingen sowohl das Inspektionspersonal bezüglich Verfügbarkeit, Teamzusammensetzung, usw. als auch Richtlinien (z.B. Lesetechniken, Checklisten, Aufgabenbeschreibungen) für den jeweiligen Projekttyp bzw. die Aufgabenstellung berücksichtigt werden. Diese Unterlagen werden den Inspektoren als Arbeitspakete zur Verfügung gestellt.

   Ausgangsseitig (*P5*) ist es aufgrund der erhalten Fehlerlisten möglich, so genannte Restfehlerschätzungen, also die Anzahl der noch verbleibenden Fehler im Software-Produkt, zu ermitteln. Diese Informationen werden an das Projektmanagement zur Analyse und Beurteilung weitergeleitet.

3. Auf der ausführenden Ebene (technische Inspektionsleitung) werden die Vorgaben des Projektmanagements (durch Projekt-, bzw. QM-Pläne) und des Inspektionsmanagements (durch definierte Arbeitpakte und Vorgehensweisen) umgesetzt und die eigentliche Inspektion durchgeführt. Nach der Fehlerfindung durch die Inspektoren (*P3*) werden Teamfehlerlisten erstellt (*P4*), die dem Inspektionsmanagement zur Restfehlerschätzung, Einschätzung der Produktqualität oder als Basis für weitere Entscheidungsfindungen zur Verfügung stehen.

In diesem Kapitel wurden bisher Reviews und Inspektion als qualitätsverbessernde Maßnahmen in frühen Phasen der Softwareentwicklung vorgestellt, die jeweiligen Abläufe gezeigt und der Zusammenhang zu Projekt- und Qualitätsmanagement hergestellt. Im nächsten Abschnitt betrachten wir einige Methoden, die den Fehlerfindungsprozess unterstützen und ein kontrolliertes Vorgehen bei der Fehlersuche ermöglichen.

5.4 Methoden zur Unterstützung von Reviews und Inspektionen

Nachdem die notwendigen Eingangsbedingungen in definierter Weise vorliegen bzw. erfüllt sind und die jeweiligen Arbeitspakete zusammengefasst wurden, können die Gutachter bzw. Inspektoren mit der Fehlersuche beginnen. Auch die Fehlersuche muss in kontrollierter Weise durchgeführt werden um systematisch beispielsweise alle Fehlerarten abdecken zu können. Dieses Kapitel beschäftigt sich mit einem Ansatz zur Fehlersuche, den so genannten *Lesetechniken*, die den Inspektor bei der Fehlersuche unterstützen. Diese Dokumente bzw. Fragenkataloge werden

allerdings als integraler Bestandteil der Arbeitspakete angesehen und den Inspektoren vor dem Review- oder Inspektionsprozess übergeben.

Charakteristiken von Lesetechniken

Um die Effizienz eines Inspektionsteams zu verbessern, müssen Methoden etabliert werden, die den Inspektionsteilnehmern den Umgang mit Dokumenten erleichtern und den Fehlerfindungsprozess unterstützen. Lesetechniken, d.h. wie man ein Softwaredokument lesen soll, können genutzt werden, um die Inspektoren zu unterstützen. Eine Lesetechnik wird als eine Reihe von Schritten bzw. als systematische Vorgehensweise definiert, die dem Inspektor als Anleitung für die Bearbeitung des Softwareprodukts dient. Durch die Anwendung einer Lesetechnik erhält der Inspektor einen besseren Zugang und ein tieferes Verständnis des zu betrachtenden Softwareprodukts und ist somit in der Lage Fehler effizienter und effektiver zu finden. Es ist jedoch die Aufgabe des Inspektors, Nutzen aus den Vorteilen einer Lesetechnik zu ziehen [Laitenberger et al., 1999].

Laitenberger unterscheidet folgende Charakteristika von Lesetechniken [Laitenberger, 2000; Schlich, 2002]:

- *Anwendungsspektrum*. Die ausgewählte Lesetechnik muss für die jeweilige Anwendung bzw. das zu inspizierende Dokument geeignet sein. Abhängig von der Struktur der Lesetechnik können spezielle Dokumentarten, also Anforderungsdokumente, Code, usw. bevorzugt werden.
- *Anleitung*. Bei den meisten Lesetechnikarten müssen spezielle Anleitungen erstellt werden, die auf die jeweiligen Anwendungsbereiche abgestimmt werden müssen. Beispielsweise sind bei verteilten Systemen andere Schwerpunkte zu setzen als bei Echtzeitsystemen. Die Anleitung unterscheidet sich ebenfalls bei Anforderungsdokumenten und Code.
- Die *Anpassbarkeit* bezeichnet die Fähigkeit, dieselbe Lesetechnik auf verschiedene Produkte anzuwenden.
- Die *Wiederholbarkeit* bezeichnet die Fähigkeit, dieselbe Lesetechnik auf dasselbe Produkt in derselben Weise anzuwenden.
- *Überdeckung*. Ein wichtiger Aspekt ist die Überdeckung, d.h. durch die Anwendung der Lesetechnik können alle Fehler im Produkt gefunden werden.
- Unter *Überlappung* versteht man die Fähigkeit der Lesetechnik die Überlappung des Fehlerauffindens zu vermeiden.

Unter Berücksichtigung der charakteristischen Eigenschaften der Lesetechniken ist es die Aufgabe des Projektleiters bzw. des Moderators, geeignete Lesetechniken für die Inspektion auszuwählen, entsprechend zu adaptieren und den Inspektoren zur Verfügung zu stellen.

In diesem Abschnitt werden die Gruppen von Lesetechniken im Überblick vorgestellt. Detaillierte Informationen zu den Lesetechniken werden in *Kapitel 12* beschrieben.

Lesetechniken im Überblick [Visek, 2003]

Die einfachste aller Lesetechniken ist die *ad-hoc Methode*, bei der ohne explizite Leseanleitung einfach nach Fehlern gesucht wird. Diese Technik ist zwar auf alle Produktionsarten anwendbar, kann aber, aufgrund der unterschiedlichen Zugänge der Inspektoren und der fehlenden Anleitungen, nicht wiederholt werden.

Checklisten-basierte Lesetechniken sind Vertreter der so genannten *nicht-systematischen* Lesetechniken, da sie das gesamte Dokument betreffen und keine Fehlerklassen bevorzugen. Diese Checklisten bestehen aus einer Reihe von Fragen mit folgendem Fokus:

1. *Allgemein gültige Fragen* beziehen sich beispielsweise auf Vollständigkeit, Konsistenz und Layout, usw.
2. *Spezifische Fragen* sind an die jeweilige Anwendungsdomäne angepasst und sollen mögliche Probleme im Zusammenhang mit dem Einsatz des Softwareprodukts aufdecken (z.B. finanzwirtschaftliche Belange für ein Projekt im Bankensektor).

Die Fragen beziehen sich dabei auf mögliche Fehlerquellen im Softwaredokument. Die Aufgabe der Inspektoren ist es, diese Frage vor, während und nach dem Lesen des Softwaredokuments zu beantworten und entsprechend zu dokumentieren. Ein

Szenario-basierte Lesetechniken sind Vertreter der *systematischen* Lesetechniken, da verschiedene Sichtweisen (Rollen), Fehlerklassen, Dokumentteile usw. im Mittelpunkt der Untersuchung stehen. Dabei werden den Inspektoren sowohl die Vorgangsweise als auch die wichtigen zu beachtenden Kriterien vorgegeben. Die Inspektoren erhalten konkrete Anleitungen (so genannte Szenarien), die wichtige Punkte beinhalten. Laut Visek gibt es aktuell die *fehlerklassen*- und *perspektiven-basierten* Ansätze. Während die fehlerklassen-basierten Szenarien die Überprüfung hinsichtlich bestimmter Fehlerklassen erlauben, ermöglicht perspektiven-basiertes Lesen die Analyse der Dokumente aus der Perspektive bestimmter Nutzergruppen [Visek, 2003]. Die Inspektoren versetzen sich also in die Lage einer Rolle und untersuchen das Dokument hinsichtlich der entsprechenden, für die jeweilige Rolle definierten Aspekte. Mögliche und in der Praxis häufig verwendete Rollen sind beispielsweise die Designer-, Tester- und Anwenderrolle [Biffl, 2001]. Bei szenario-basierten Lesetechniken muss darauf geachtet werden, dass durch die Anwendung der Lesetechnik auch alle möglichen Fehler tatsächlich gefunden werden können. Die Aufgabenpakete müssen daher entsprechend gestaltet sein.

Während Reviews und Inspektion in allen Phasen, speziell auch in frühen Phasen, der Softwareentwicklung angewandt werden können, ist die *Methode Testen* im Normalfall erst möglich, wenn es Module bzw. Programmteile gibt.

5.5 Methode Testen

Eine zentrale Rolle im Spektrum der Qualitätssicherung ist das Testen, also das Finden von Fehlern am fertigen Produkt, in diesem Fall eines Programms bzw. einer Komponente.

Testen bezeichnet allgemein die Tätigkeit, ein Softwareprogramm auszuführen, um dabei Fehler zu finden. Ein *Fehler* ist in diesem Zusammenhang jede Abweichung des tatsächlichen Verhaltens vom Sollverhalten des Systems [Frast, 2002].

Klassifikation	Maßnahmen
analytische oder fehleridentifizierende Maßnahmen	
• dynamische Maßnahmen, d.h. das Objekt wird ausgeführt	Testen
• statische Maßnahmen, d.h. das Objekt wird nicht ausgeführt	Review
	formale Verifikation
	Messung von Qualitätsindikatoren
	…
konstruktive oder fehlervermeidende Maßnahmen	Verwendung von Hochsprachen
	Designmethoden

Tabelle 5.9: Einordnung des Testens in die Maßnahmen der Qualitätssicherung [Mertens, 1997]

Das Sollverhalten kann durch explizite oder implizite Vorgaben in entsprechenden Anforderungs- oder Designdokumenten definiert sein und wird durch Testen verifiziert[1] bzw. validiert[2]. Daher ist die Methode Testen, angewandt auf das fertige Produkt oder die fertige Komponente, die wichtigste und am häufigsten angewandte Methode der Qualitätssicherung.

Im Rahmen des Software-Entwicklungsprozesses kommen zahlreiche verschiedene Testebenen zum Einsatz. Die folgende Aufzählung gibt die wichtigsten Ansätze bzw. Testebenen wieder:

- Beim *Modultest* werden einzelne Module gegen ihre Designvorgaben getestet. Bilden mehrere Module eines Systems eine größere Einheit, so sollte der *Integrationstest* Fehler im Zusammenwirken der Module aufdecken.
- Ein *Systemtest* prüft das Gesamtsystem gegen die Spezifikation. Bei Softwareprodukten für den anonymen Markt unterscheidet man ferner zwischen Alpha- und Beta-Test.
- Beim *Alpha-Test* arbeiten Anwender in der Umgebung des Herstellers.
- Dagegen erproben beim *Beta-Test* ausgewählte Kunden das Produkt in der eigenen Umgebung, bevor es für den Markt freigegeben wird.

Abhängig von der Testebene und dem aktuellen Anwendungsfall verfolgen Tests unterschiedliche Ziele:

---

[1] *Verifikation*: Unter einer Verifikation versteht man die Überprüfung eines Softwareprodukts im Hinblick auf die direkten Vorgabedokumente, z.B. ein implementiertes Modul bezüglich der Modulspezifikation („das Produkt setzt die Anforderungen richtig um“).

[2] *Validierung*: Unter einer Validierung versteht man die Überprüfung eines Produktes im Hinblick auf die Anforderungen bzw. den Kundenwunsch („das Produkt setzt die richtigen Anforderungen um“).

- Im *Funktionstest* werden Abweichungen der tatsächlichen von der geforderten Funktionalität gesucht.
- Der *Lasttest* forscht nach Abweichungen des geforderten vom tatsächlichen Lastverhalten.

Seit langem weiß man, dass Fehler umso höhere Kosten verursachen, je mehr Zeit zwischen der Entstehung des Fehlers und seiner Entdeckung vergeht. Aus diesem Grund sollte das Testen entsprechend der Entwicklung in Testphasen organisiert werden, die zeitlich und sachlich abgegrenzte Abschnitte des Testens darstellen.

Fehler in Anforderungsdokumenten oder Entwürfen können nicht durch Testen, sondern durch statische Prüfmethoden, Simulationen oder Prototyping erkannt werden. Daher gehört zu einer systematischen, effektiven und effizienten Qualitätssicherung ein umfassendes System von *phasenbezogenen Prüfmaßnahmen*. Diese Prüfmaßnahmen müssen aufeinander abgestimmt sein, sodass es möglich ist, alle Arten von Abweichungen (ohne Überschneidungen) zu identifizieren.

Je nachdem ob beim Testen die innere Struktur der Software berücksichtigt wird oder nicht, spricht man von *Whitebox*- oder *Blackbox*-Testen.

- Beim *Blackbox-Testen* wird die Funktionalität der Software überprüft, in dem man definierte Eingabedaten verwendet, die Soll-Ausgaben mit den Ist-Ausgaben vergleicht und Übereinstimmung oder Abweichung dokumentiert. Beispielsweise kann man beispielsweise nach der Äquivalenzklassenmethode Testfälle so bestimmen, dass die verschiedenen Eingabedaten, die zu einem Testfall gehören, äquivalent verarbeitet werden. Als Testdaten wählt man z. B. nach der Grenzwertmethode in jeder Äquivalenzklasse Grenzwerte, wie die leere Eingabe für eine Texteingabe oder extrem große oder kleine Eingabewerte.
- Beim *Whitebox-Testen* wir die innere Struktur des Programms untersucht, in dem – abhängig von den Eingabedaten – die jeweiligen Programmwege durchlaufen werden. Der Test ist erfolgreich, wenn alle erwarteten Wege durchlaufen werden. Man unterscheidet man *kontrollflussorientierte* und *datenflussorientierte* Methoden. Bei kontrollflussorientierten Methoden versucht man, Testdaten so zu wählen, dass möglichst viele unterschiedliche Wege des Kontrollflusses ausgeführt werden. Bei datenflussorientierten Methoden versucht man, die Testfälle so zu bestimmen, dass nahezu alle Variablen die Wege von der Wertzuweisung bis zur Verwendung durchlaufen.

5.6 Zusammenfassung

Dieses Kapitel behandelt die wesentlichen Aspekte der Qualitätssicherung und stellt die Zusammenhänge zum Qualitätsmanagement und einigen Methoden und Werkzeugen der Qualitätssicherung her. Die Qualitätssicherung (QS) ist für die Umsetzung der Vorgaben des Qualitätsmanagements, für die Auswahl, Steuerung und Überprüfung zuständig und bildet somit die Schnittstelle zwischen Management und operativer Umsetzung.

Qualitätsfaktoren und Beurteilung. Je nach Sichtweise, Rolle und Anwendung existieren Anforderungen an das Produkt oder die Dienstleistung, die allgemein als Qualitätsfaktoren bezeichnet werden. Allgemeine Qualitätsfaktoren werden beispielsweise von Normungsinstituten vorgestellt und weiterentwickelt (z.B. IEEE). Das Ziel ist es, diese Qualitätsfaktoren für die

entsprechenden Produkte zu erreichen. Dieses Vorhaben bedingt allerdings die Beurteilung dieser Faktoren und somit auch die Messung und Auswertung von Daten. Dazu werden Qualitätsfaktoren in Qualitätsmodellen klassifiziert und mit entsprechenden Metriken zur Bewertung bestückt. Die Entwicklung solcher Modelle (z.B. Qualitätsmodell nach McCall), Metriken und deren Zusammenspiel wird beispielsweise durch die Goal-Question-Metric (GQM) ermöglicht.

Ablaufmodell der Qualitätssicherung. Um qualitativ hochwertige Software produzieren zu können, ist es nicht ausreichend, einfach nur Vorgehensmodelle zu verwenden und anschliessend QualitätsMaßnahmen anzuwenden. Vielmehr ist die Softwareentwicklung ein Zusammenspiel vieler unterschiedlicher Disziplinen, wie es beispielsweise am V-Modell des Bundes visualisiert wurde. Dieses Modell besteht u.a. aus einem "Submodell Qualitätssicherung", das in diesem Kapitel erklärt wurde.

Reviews und Inspektionen. Wie im Ablaufmodell der Qualitätssicherung gezeigt wurde, sind zahlreiche Überprüfungen der Produkte und Zwischenprodukte notwendig um qualitativ hochwertige Produkte herstellen zu können. Diese Überprüfungen müssen in jeder Phase der Softwareentwicklung umgesetzt werden. Reviews und Inspektionen sind gut geeignet, um diese Überprüfungen bereits in frühen Phasen der Softwareentwicklung für alle Arten von Softwareprodukten, beispielsweise Anforderungsdokumente, durchzuführen. In diesem Kapitel wurde auf verschiedene Arten und Typen bzw. auf allgemeine Vorgangsweisen eingegangen.

Methoden zur Unterstützung von Reviews und Inspektionen. Das Hauptziel von Reviews und Inspektionen ist die Fehlerfindung in Softwareprodukten. Um dieses Ziel zu erreichen, sind Methoden notwendig, die es erlauben, strukturiert und kontrolliert nach Fehlern zu suchen. Lesetechniken sind ein Ansatz, um die Effizienz und Effektivität der Fehlerfindung in Softwaredokumenten zu erhöhen. Dieses Kapitel gibt eine kurze Einführung in die Thematik "Lesetechniken".

Methode Testen. Eine weit verbreitete Methode zur Fehlerfindung sind Testmethoden. Sie kann allerdings erst sinnvoll angewandt werden, wenn fertige Programme bzw. Komponenten vorliegen. Den Abschluss dieses Kapitels bildet eine grobe Einführung in die Thematik "Testen" für die folgenden Kapitel.

5.7 Literaturverweise

[Basili et al, 1994] Basili, Victor R.; Caldiera, Gianluigi; Rombach, H. Dieter: "The Goal Question Metric Approach"; Encyclopedia of Software Engineering - 2 Volume Set, pp 528-532; John Wiley & Sons, Inc.; 1994.

[Basili et al., 1996] Basili, Victor R.; Briand, Lionel C.; Melo, Walcelio L.: „*A Validation of Object-Oriented Design Metrics as Quality Indicators*", 1996, IEEE Transactions on Software Engineering, Vol 22, No. 10, 751-761.

[Biffl et al., 2000] Biffl, Stefan; Freimut, Bernd; Laitenberger, Oliver: „*Investigating the Cost-Effectiveness of Reinspection in Software Development*", 2000, Technical Report 00-25; Dept. Software Engineering, Vienna.

[Biffl et al., 2001] Biffl, Stefan; Gutjahr, Walter: „Influence of Team Size and Defect Detection Technique on Inspection Effectiveness", 2001.

[Biffl, 2001] Biffl, Stefan: „*Software Inspection Techniques to support Project and Quality Management*", Shaker Verlag, 2001, ISBN: 3-826-58512-7.

[Bisant et al, 1989] Bisant, D.B. and Lyle, J. R.: „A Two-Person Inspection Method to improve Programming Productivity", in IEEE Transactions on Software Engeneering, Vol. 15, No. 10, pp 1294-1304, 1989.

[Fagan, 1976] Fagan, M.E: „*Design and Code Inspections to Reduce Errors in Program Development*", 1976, IBM System Journal, No 3; p182-211.

[Frast, 2002] Frast, Denis: "Testautomatisierung für kommerzielle Softwaresysteme - Eine wiederverwendbare Architektur für Testautomatisierung"; 2002; Diplomarbeit, TU Wien.

[Gilb et al., 1993] Gilb, Tom; Graham, Dorothy: „*Software Inspection*", Addison-Wesley, 1993, ISBN: 0-201-63181-4.

[Hansen, 1998] Hansen, Hans Robert: "Wirtschaftsinformatik I", 7. Auflage, Verlag UTB, 1998; ISBN: 3-825-20802-8.

[Hindel, 1996] Hindel, Bernd: "Qualität ist messbar: Software-Metriken"; Design & Elektronik Nr. 24; November 1996.

[IEEE, 610-1990] IEEE Standard Glossary of Software Engineering Terminology, 1990.

[IEEE, 1028-1997] IEEE Standard for Software Reviews, 1997.

[IEEE, 1990] IEEE-STD 610.12-1990 - Standard Glossary of Software Engineering Terminology, 1990.

[IESE, 2003] Fraunhofer Institut, "Submodell Qualitätssicherung"; http://www.iese.fhg.de/VModell.

[Kitchenham et al, 2003] Kitchenham, B; Pfleeger, S.L.: "Software Quality: The Elusive Target"; IEEE 1996; 2003.

[Klaeren, 2003] Klaeren, Herbert: "Softwaretechnik – Vorlesungsmanuskript"; Universität Tübingen; 2003.

[Knight et al, 1993] Knight, J. C. and Myers, E. A. :"An improved Inspection Technique", in *Communications of the ACM* ,Vol. 36,No. 11, pp. 51-61, 1993.

[Laitenberger, 1998] Laitenberger, Oliver: „*Studying the Effects of Code Inspection and Structural Testing on Software Quality*", 1998, IESE-Report 024.98/E, May 1998; Fraunhofer Institute for Experimental Software Engineering, Germany, ISERN-98-10.

[Laitenberger et al., 1999] Laitenberger, Oliver; Atkinson, Colin; Schlich, Maud; El Emam, Khaled: „*An Experimental Comparison of Reading Techniques for Defect detection in UML design Documents*", 2000, IESE-Report 080.99/E, December 1999; INSERN, 2000-01.

[Laitenberger, 2000] Laitenberger, Oliver: "Cost-Effective Detection of Software Defects Through Perspective-based Inspections"; PhD Theses in Experimental Software Engineering Vol. 1, 2000.

[Martin et al, 1990] Martin, J.; Tsai, W. T.: „N-Fold Inspection: A Requirements Analysis Technique", in *Communications of the ACM* ,Vol. 33,No. 2, pp. 225-232, 1990.

[Mertens, 1997] Mertens, Peter et al.: „Lexikon der Wirtschaftsinformatik“, 3. Auflage; Springer; 1997; ISBN: 3-540-61917-8.

[Parnas et al, 1985] Parnas, D.L.; Weiss, D.M.:“ Active Design Reviews: Principles and Practices“, in *Proceedings of the 8th International Conference on Software Engineering,* pp 132-136, 1985.

[Schlich, 2002] Schlich, Maud: "Inspektion des Systemlastenhefts"; IESE-Report Nr. 36.02; Juli 2002.

[Thaller, 2000] Thaller, Georg Erwin: „*Software Qualitat*“, VDE Verlag, 2000, ISBN: 3-800-72494-4

[VModell, 1997] V-Modell: Entwicklungsstandard für IT-Systeme des Bundes, 1997, http://www.iese.fhg.de/VModell/.

[Visek, 2003] Virtuelles Software-Engineering Kompetenzzentrum; http://www.visek.de.

[Winkler, 2003] Winkler, Dietmar: "Empirical Evaluation of Inspections as a Quality Management Approach for Defect Detection in the Context of Software Engineering"; 2003; Diplomarbeit, TU Wien.

[Zilahi-Szabo, 1998] Zilahi-Szabo, Miklos Geza: “Grundzüge der Wirtschaftsinformatik”, Oldenburg Verlag, 1998, ISBN: 3-486-25516-9.

[Zuser et al., 2001] Zuser, Wolfgang; Biffl, Stefan; Grechenig, Thomas; Köhle, Monika: „*Software Engineering mit UML und dem Unified Process*“, Pearson Studium, 2001, ISBN: 3-827-37927-2.

5.8 Übungen und Fragen

1. Beschreiben Sie mindestens 5 Qualitätsfaktoren und überlegen Sie sich mögliche Metriken um diese Qualitätsfaktoren messen zu können.
2. Wie definiert die IEEE die Qualitätsfaktoren?
3. Beschreiben Sie kurz den Goal-Question-Metrik Ansatz.
4. Welche Vorteile sind mit dem V-Modell des Bundes verbunden und welche Zusammenhänge bestehen zwischen analytischen und konstruktiven Maßnahmen?
5. Beschreiben Sie das Submodell "Qualitätssicherung" im V-Modell und berücksichtigen Sie dabei die jeweiligen Produkte, die in den jeweiligen Phasen erstellt werden.
6. Welche Arten von Reviews kennen Sie? Wann werden sie eingesetzt?
7. Welche Rollen sind bei Reviews / Inspektionen zu beachten?
8. Erklären Sie den Ablauf von Reviews bzw. Inspektionen.

9. Sie haben für die Durchführung von Reviews 4 Personen zur Verfügung. Wie besetzen Sie die einzelnen Rollen? Argumentieren Sie, warum Ihre Besetzung der Reviewrollen sinnvoll ist.

10. Diskutieren Sie die technischen Dimensionen von Software Inspektionen.

11. Erläutern Sie die Zusammenhänge von Qualitätsmanagement und Inspektionen.

12. Erläutern Sie die Zusammenhänge zwischen Qualitätsmanagement und Qualitätssicherung.

13. Welche Charakteristiken sind bei der Auswahl der Lesetechnik zu beachten?

14. Erklären Sie kurz die unterschiedlichen Testebenen, die im Rahmen des Softwareentwicklungsprozesses zum Einsatz kommen.

15. Erläutern Sie den grundlegenden Unterschied zwischen Blackbox und Whitebox-Testfällen?

16. Welche Qualitätsmaßnahmen können in welchen Phasen des V-Modells eingesetzt werden?

17. Sie sind als Qualitätsleiter für ein Softwareprojekt in der Grössenordnung von 6 Personenjahren zuständig. Aktuell sind 10 Softwareentwickler aus allen relevanten technischen Bereichen in das Projekt involviert. Das Projekt befindet sich am Ende der Designphase, ist allerdings stark im Zeitverzug. Aufgrund dieses Zeitdrucks will der Projektleiter auf Reviews verzichten. Sie sind jedoch der Ansicht, dass aufgrund der kritischen Anwendungsdomäne Reviews unbedingt durchgeführt werden müssen? Wie setzen Sie gegenüber dem Projektleiter die Notwendigkeit von Reviews durch. Wie gehen Sie bei der Organisation der Reviews vor?

18. Sie sind Moderator eines Reviewprozesses. Welche Schritte sind unter Betrachtung des gesamten Prozesses notwendig, um eine Review erfolgreich durchzuführen?