Prozessorientiertes Produktqualitätsmonitoring für die Entwicklung elektronischer Systeme

Dissertation zur Erlangung des Grades eines
Doktors der Ingenieurswissenschaften

Vorgelegt von

Dipl. Inform. Stefan Häusler

Gutachter

Prof. Dr.-Ing. Axel Hahn

Prof. Dr.-Ing. Wolfgang Nebel

Prof. Dr.-Ing. Roland Jochem (TU Berlin)

Datum der Disputation

24.1.2012

Danksagung

Als erstes danke ich Prof. Dr. Axel Hahn für die Betreuung dieser Dissertation und der Möglichkeit in verschiedenen industriellen Forschungsprojekten an der Thematik mitzuarbeiten. Ich danke ebenfalls den Prof. Dr. Wolfgang Nebel und Prof. Dr. Roland Jochem für die konstruktive Kritik und die Begutachtung dieser Arbeit.

Ein Großteil der Arbeit wurde am OFFIS Institut für Informatik unterstützt, an dem ich an den Projekten PRODUKTIV+ und CESAR mitarbeiten durfte. Ich danke allen beteiligten Projektpartnern für die vielen Diskussionen und Kommentare. Ein besonderer Dank geht dabei an die Projektpartner der Robert Bosch GmbH Abteilung in Reutlingen, die mir die Durchführung eines der beiden Evaluierungsbeispiele ermöglicht haben. Weiterhin danke ich allen Kollegen des Bereichs Verkehr am OFFIS – Institut für Informatik und der Arbeitsgruppe von Prof. Axel Hahn an der Universität Oldenburg mit denen ich über die Jahre zusammen arbeiten durfte. Die Einblicke und Diskussionen in diverse Bereiche des Elektronikentwicklungsprozesses trugen ohne Zweifel zur Lösungsfindung bei.

Ich danke meinen Eltern für die Jahrzehnte der Unterstützung. Ohne sie wäre das erreichte nicht möglich gewesen. Vor allem Danke ich ihnen für das Vertrauen in mich und meine Art Dinge anzugehen und Entscheidungen zu treffen.

Vor allem danke ich jedoch meiner Freundin Mareen für ihre Geduld und Verständnis die sie ohne Zweifel während der jahrelangen Arbeit aufbringen musste. Ich danke ihr für die Motivation und Unterstützung in schlechten Phasen der Arbeit, auch wenn es für sie nicht immer so aussah, als hätte es viel geholfen. Ich danke ihr und unseren Sohn Jonas für die Freuden des Lebens und die notwendige Ablenkung, ohne die ich vielleicht nicht durchgehalten hätte.

Zusammenfassung

Auf Grund ständig steigender Produktkomplexität ist neben den damit verbundenen technischen Herausforderungen ebenfalls das Planen, Managen und Monitoring von Entwicklungsprojekten in der Elektronikentwicklung herausfordernder denn je. Insbesondere die Qualitätssicherung ist hier ein elementarer Bestandteil. Unternehmen setzen eine Vielzahl an heterogenen Verifikationsmethoden in Kombination ein, die jedoch für sich nur einzelne Aspekte betrachten und keine Gesamtsicht auf den Qualitätsstand liefern. Diese wird jedoch benötigt, um effektiv Aussagen über die Gesamtqualität und damit den Projektstatus zu treffen. Dies scheitert jedoch entweder an einer fehlenden einheitlichen Qualitätsdefinition, dessen unzureichenden Integration mit aktuellen Entwicklungsprozessen oder der ineffizienten Anpassung unternehmensweiter Qualitätsdefinitionen auf einen konkreten Projektkontext.

Die Arbeit realisiert diese Gesamtsicht und entwickelt ein prozessorientiertes Produktqualitätsmonitoring für die Elektronikentwicklung. Ausgehend von einer prozessorientierten Qualitätsdefinition entlang eines Prozessmodells erfolgt eine integrierte Bewertung anfallender Verifikation & Validierungsergebnisse (kurz V&V). Zur Unterstützung von Prozesskontrolle und -steuerung werden diese Daten entsprechend definierter Prozessziele interpretiert. Gleichzeitig ermöglicht die prozessorientierte Qualitätsdefinition deren effiziente Anpassung auf ein konkretes Projekt begleitend zur eigentlichen Prozesskalibrierung.

Technische Grundlage der Methodik ist ein Modellierungsframework bestehend aus Metamodellen zur Beschreibung von Anforderungen, Design und V&V Daten, sowie existierenden Anforderungsverfolgungskonzepten, welche die semantische Basis für die Qualitätsbewertung bilden. Für eine effektive Planung, Überwachung und Steuerung der Qualitätssicherung in Projekten verknüpft die Arbeit diese Produktmetamodelle mit einem Prozessmetamodell, welche die Definition von Prozessen, zugehörigen Qualitätsmetriken und -zielen ermöglicht.

Die Arbeit leistet folgende wissenschaftliche Beiträge: (1) Eine umfassende Analyse und Beurteilung der gegenwärtig gegebenen Methoden zur Qualitätsbewertung in der Elektronikentwicklung, (2) Motivation und Konzeption eines Vorgehens für ein prozessorientiertes Produktqualitätsmonitorings sowie (3) die vollständige Evaluation der daraus resultierenden Möglichkeiten im Kontext zweier Beispiele (Hardwareentwicklung, Entwicklung sicherheitskritischer eingebetteter Systeme).

Inhaltsverzeichnis

1 Einleitung

Elektronik bzw. eingebettete Systeme sind heutzutage Bestandteil in nahezu allen Bereichen des heutigen Lebens. Ohne sie würde kein Flugzeug fliegen, kein Auto fahren, noch würden Fabriken funktionieren. Elektronik ist einer der zentralen Innovationstreiber. Im Automobil bestimmen z.B. eingebettete Systeme bis zu 90% aller Innovationen (Fast, 2005). Auch bei der Lösung zukünftiger gesellschaftlicher Herausforderungen, wie der Reduzierung des Stromverbrauchs durch intelligente Mess- und Steuergeräte, dem Energiemanagement im Automobil zur Reduzierung von Treibstoffverbrauch und Emissionen oder auch bei der Gestaltung altersgerechter Lebenswelten (siehe z.B. (Damm, 2009a)), spielen eingebettete Systeme eine zentrale Rolle.

Diese Rolle stellt hohe Qualitätsansprüche an die Systeme. Gerade in der Energieversorgung, der Gesundheit oder auch der Mobilität führen Ausfälle und Fehler anders als vielleicht in einem Handy zu potentiell schwerwiegenden Konsequenzen. Ziel dieser Arbeit ist es, Unternehmen dabei zu unterstützen, qualitativ hochwertige Elektroniksysteme zu entwickeln. Dieses einleitende Kapitel trägt dazu durch die Detaillierung der Problem- und Zielstellung bei. Ausgehend von der Motivation und Problemstellung in Abschnitt 1.1 und 1.3 formuliert Abschnitt 1.4 Ziele und Beitrag der vorliegenden Arbeit. Zum Verständnis führt dieses Kapitel bereits in Abschnitt 1.2 grundlegende Begriffe ein. Der abschließende Abschnitt skizziert den weiteren Aufbau dieser Arbeit.

1.1 Motivation

Die Produktentwicklung ist ein überlebenswichtiger und herausfordernder Prozess. Überlebenswichtig, da im Zuge der Globalisierung der Wert erfolgreicher, vermarktbarer Innovation steigt und nur wenige Unternehmen auf die Entwicklung neuer Produkte verzichten können. Herausfordernd auf Grund ihrer Komplexität, so dass ihr Management trotz der hohen Bedeutung und der immensen Anstrengungen mit zahlreichen Unwägbarkeiten behaftet ist und Projekte überdurchschnittlich oft scheitern. Je nach Branche und Innovationsgrad des betrachteten Vorhabens lässt sich anhand zahlreicher Studien wie z.B. (Buschermöhle, 2006), (Standish Group, 2004), (GPM, 2004), (Hartmuth, 2003), (Feldmuller, 2006), (Mandl-Stiegnitz, 1999), (Commes, 1997) die grobe Einschätzung treffen, dass ca. ein Drittel aller Entwicklungsprojekte scheitern (Strickmann, 2009). Wie die Ergebnisse aus (Standish Group, 2004) (dargestellt in Abbildung 1) illustrieren, ist nur ein Drittel der Entwicklungsprojekte erfolgreich (d. h. Abschluss unter Einhaltung aller finanziellen, zeitlichen und qualitativen Ziele), die übrigen Projekte verfehlen ein oder mehrere Teilziele.

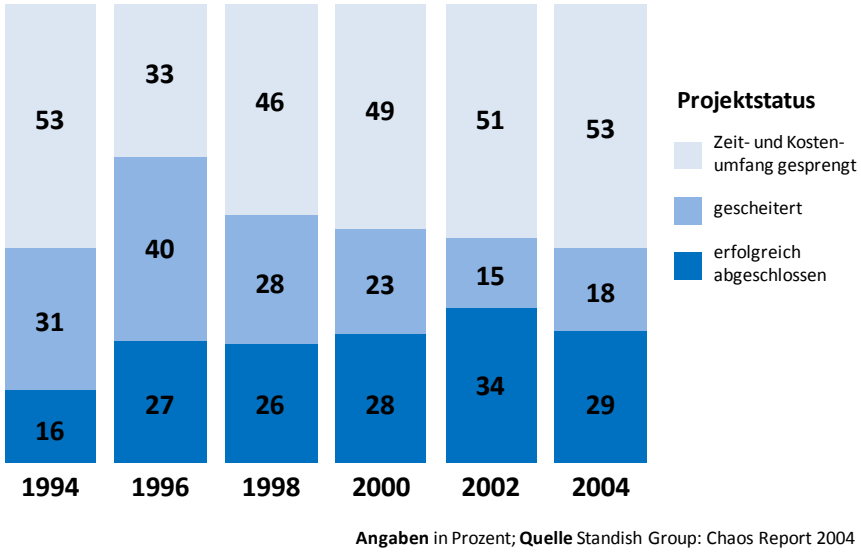

Abbildung 1: Ergebnisse des CHAOS-Reports (Standish Group, 2004).

Auch wenn die Studien unterschiedliche Branchen und Domänen betrachten, so lassen sich diese beiden Eigenschaften insbesondere auf die Elektronikentwicklung übertragen, die sich durch ihren hohen Innovationsgrad und ihre stark gestiegene Komplexität auszeichnet. Ein Trend, der auch heute noch nicht am Ende angelangt ist. Verschiedene Randbedingungen verstärken diesen Faktor und bilden vor allem in ihrer Kombination die Herausforderung der heutigen Elektronikentwicklung. Entwicklungsprojekte müssen von engen Zeitfenstern („time to market“) über die Integration verschiedener Fachrichtungen bis zu höchsten Qualitätsansprüchen zahlreiche Anforderungen erfüllen. Hinzu kommen monetäre und personelle Restriktionen, die Notwendigkeit zur Berücksichtigung des gesamten Produktlebenszyklus sowie eine Vielzahl verwendeter Werkzeuge.

Die Elektronikentwicklung ist ein vorrangig Informationen verarbeitender Prozess, welcher gerade in den frühen Entwicklungsphasen unsichere und mit Annahmen behaftete Anforderungen als Ausgangsbasis hat. In der Elektronikentwicklung verfeinern und ändern Entwickler diese schrittweise entlang mehrerer Entwicklungsphasen, von der System- bis hin zu Hard- und Softwareentwicklung. Parallel entsteht hierzu eine stetig konkretisierte Produktbeschreibung: von der Systemarchitektur über Hard- und Softwarearchitekturen der einzelnen Komponenten bis hin zu deren Implementierung. Sie setzen die definierten Anforderungen um. Um Probleme in Entwicklungsprojekten frühzeitig zu erkennen und Fehlschläge zu minimieren, bedarf es einer objektiven und regelmäßigen Qualitätsbewertung, beginnend bereits in den frühen Phasen eines Projekts. Denn Qualitätsprobleme sind eines der Hauptgründe für gescheiterte Projekte. Diese finden entweder ihren Weg in das Produkt, Qualitätsziele werden nicht erreicht, oder führen zu Mehraufwand in Zeit und Kosten, um die Probleme zu beheben. Eine entwicklungsbegleitende Qualitätsüberwachung schafft Transparenz und liefert Antworten auf operative projektbezogene Fragestellungen: Schaffe ich meine Meilensteintermine? Wie ist meine aktuelle Qualität? Welche Anforderungen sind erfüllt?

Entwicklungsleiter beschäftigen sich neben diesen operativen aber auch mit längerfristigen strategischen Fragestellungen: Welche Methoden und Werkzeuge optimieren meinen Entwicklungsprozess? Wie war die Produktivität einzelner Prozessschritte oder ganzer Projekte? Wie sehen optimale Abläufe aus? Ohne einen geeigneten Bewertungsmaßstab für Ergebnisse von Entwicklungsprozessen, wie z.B. deren Qualität, ist eine Antwort auf diese Fragen kaum möglich:

„If you can't measure something, you can't understand it. If you can't understand it, you can't control it. If you can't control it, you can't improve it“(vgl. (Harrington, 1991))

Ohne Messen, keine Kontrolle. Ohne Kontrolle, keine Verbesserung. Auch Prozessreifegradmodelle, wie dem Capability Maturity Model Integration kurz CMMI (CMMI, 2010) untermauern dieses Motto. CMMI behandelt die systematische und disziplinübergreifende Betrachtung der Wertschöpfungs- und vor allem auch unterstützenden Prozesse. Im Standard heißt es „Die Qualität eines Systems oder Produkts wird entscheidend durch die Qualität seiner Entwicklungsabläufe beeinflusst“. CMMI bewertet dabei einzelne sogenannte Prozessgebiete nach Fähigkeitsgraden und die gesamte IT-Entwicklungsorganisation über mehrere Prozessgebiete hinweg nach Reifegraden. CMMI umfasst insgesamt fünf Reifegradebenen (siehe Abbildung 2). Je höher der Reifegrad eines Unternehmens, desto fähiger ist sie - nach CMMI - bei der Abwicklung von Entwicklungsprojekten. Bei CMMI handelt es sich dabei nicht um ein Vorgehensmodell für die Entwicklung selbst. Es beschreibt in Form von Zielen und bewährten Praktiken für einzelne Prozessgebiete lediglich „Was“ zu tun ist. Es soll Organisationen zur stetigen Prozessverbesserung führen. CMMI fordert daher die Bestimmung des „Wie“ von der Organisation selbst. Ein Unternehmen, welches Methoden für das quantitative Projektmanagement und zur Bewertung von Prozessleistung in seinen Entwicklungsprozess einführt, erfüllt ein Kriterium des vierten Prozessreifegrads, die detaillierte Ausgestaltung ist jedoch nicht Inhalt von CMMI.

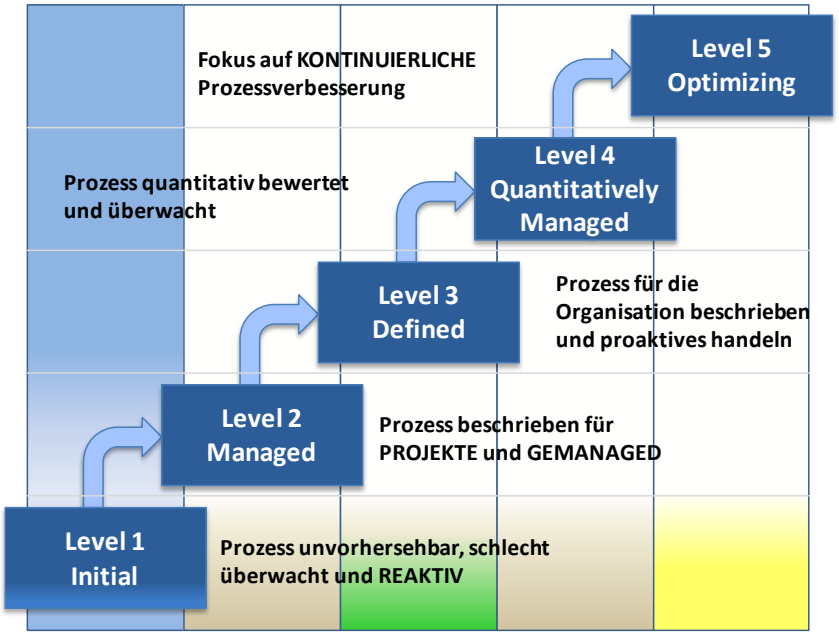

Abbildung 2: CMMI Reifegradmodell

Basierend auf den geschilderten Problemen in der Produktentwicklung und Prozessreifegradmodellen wie CMMI, welches anerkannt als ein Qualitätssiegel „reifer“ Entwicklungsprozesse gilt, identifiziert die vorliegende Arbeit zwei allgemeine Herausforderungen:

1. Entwicklungsbegleitendes Qualitätsmonitoring, welches Transparenz über den aktuellen Entwicklungsstand schafft, zeitnah Probleme identifiziert und damit ein effizientes Entwicklungsprojektcontrolling unterstützt.
2. Entwicklungsbegleitendes Qualitätsmonitoring zur Messung von Qualität als ein zentraler Indikator bei der Effizienzbewertung von Entwicklungsprozessen.

Das im Rahmen dieser Arbeit entwickelte prozessorientierte Produktqualitätsmonitoring für die Entwicklung elektronischer Systeme legt die Grundlage, um beide genannten Herausforderungen zu adressieren. Die Arbeit vertieft dabei schwerpunktmäßig die erste Herausforderung.

Im Verlauf der Einleitung führen die weiteren Abschnitte zentrale Begriffe dieser Arbeit ein, detaillieren die Probleme rund um eine entwicklungsbegleitende Qualitätsbewertung und definieren Ziele und Beiträge dieser Arbeit. Der letzte Abschnitt schließt dieses Kapitel mit einer Übersicht des weiteren Verlaufs dieser Arbeit ab.

1.2 Begriffsbildung

Dieser Abschnitt definiert zentrale Begriffe, deren Semantik für das Verständnis der vorgestellten Probleme und Konzepte grundlegend ist.

1.2.1 Qualität

Auf Grund seiner zentralen Bedeutung in dieser Arbeit erhält der Qualitätsbegriff eine separate Einführung. Dieser Abschnitt untersucht existierende Interpretationen des Qualitätsbegriffs auf ihre Anwendung im Kontext dieser Arbeit.

Bereits Garvin (Garvin, 1984) beschrieb Qualität als ein komplexes Konzept mit multiplen Facetten und definierte fünf verschiedene Perspektiven:

- Die **transcendent view** definiert Qualität als etwas was erkannt aber nicht definiert werden kann.
- Die **product view** definiert Qualität durch qualitätsrelevante Produktcharakteristika.
- Die **user view** betrachtet Qualität als subjektives Attribut, welches im Auge des Betrachters liegt.
- Die **manufacturing view** definiert Qualität als Anforderungserfüllungsgrad und
- die **value-based view**, die Qualität inhärent mit den Kosten verknüpft sieht, um diese zu erreichen.

Zwei dieser Perspektiven sind aus Sicht der Qualitätsbewertung zur Unterstützung des operativen Projektcontrollings von besonderem Interesse. Zum einen die Produktsicht (product view), welche sich auf qualitätsrelevante Produktcharakteristika fokussiert. Zum anderen die Fertigungssicht (manufacturing view), die Qualität als den Grad der Übereinstimmung von Produkteigenschaften mit seinen Anforderungen und Spezifikationen beschreibt. Die zweite Definition deckt sich mit einer Reihe weiterer, wie z.B. mit „Conformance *to requirements*" (Crosby, 1980), „*Quality is the degree to which a specific product conforms to a design or specification*" (Gilmore, 1974). Geiger bezeichnet Qualität als einen Maßstabsbegriff (Geiger, 2001). Es ist das Ergebnis eines Vergleichs zwischen der realisierten Beschaffenheit, also den aktuellen Eigenschaften des betrachteten Objekts und der geforderten Beschaffenheit. Kurzum ist Qualität die „Realisierte Beschaffenheit einer Einheit bezüglich Qualitätsforderungen an diese". Dies deckt sich im Wesentlichen mit den bereits vorher genannten Qualitätsdefinitionen. Die Qualitätsnorm ISO 9000:2005-12 definiert Qualität als „*Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt*". Dies entspricht sowohl den klassischen Definitionen wie z.B Garvin, Crosby und Gilmore als auch der Kurzdefinition nach Geiger. Den Definitionen folgend, ist eine Qualitätsbewertung anforderungsabhängig. Anforderungen sind üblicherweise unternehmens- und projektspezifisch, daher gilt dies folglich auch für die Qualitätsbewertung selbst.

Andere Qualitätsdefinitionen rücken den Benutzer stärker in den Mittelpunkt. So definiert z.B. Juran (Juran, 1988) Qualität als *„Fitness for use"* oder Weinberg als *„Quality is value to some person"* (Weinberg, 1991).

Das Total Quality Management (TQM) hat den Qualitätsbegriff ebenfalls maßgeblich geprägt. TQM geht auf die Lehren von William Edwards Deming (Deming, 2000) zurück, der ebenfalls die Bedürfnisse des Kunden als oberstes Qualitätsziel ausgibt. In seiner Managementlehre ist der Hebel zur Qualitätsverbesserung der Prozess. Dieser ist stetig zu verbessern, um schließlich qualitativ hochwertige Produkte zu erzeugen. Diese Qualitätsmanagement Philosophie wurde durch die japanische Automobilindustrie übernommen und zum Total Quality Management weiterentwickelt. Das TQM fasst insbesondere den Fokus von Qualität, also die betrachtete Einheit deutlich weiter. Nach Zollondz (siehe (Zollondz, 2006), S.174) ist Qualität im TQM *„die realisierte Beschaffenheit von Einheiten mit unmittelbaren, mittelbaren und keinem direkten Qualitätsbezug bezüglich Qualitätsforderungen und anderer Forderungen an diese Einheit"*. Der Fokus aller Mitarbeiter liegt auf totaler Kunden- und Qualitätsorientierung. Alle Unternehmensprozesse sind auf diese Ziele ausgerichtet. TQM fokussiert daher nicht nur das Produkt, sondern alle Einheiten in einem Unternehmen, wie das Management, die Mitarbeiter und die Prozesse (siehe auch (Rothlauf, 2010)).

Welche dieser Definitionen ist nun am geeignetsten, um als Grundlage für diese Arbeit zu dienen, dessen Ziel ein entwicklungsbegleitendes Produktqualitätsmonitoring ist? Eine benutzer- bzw. kundenorientierte Definition von Qualität ist gerade während der Entwicklung nur sehr schwer messbar und eignet sich dann auch nur, sobald vorführbare Prototypen existieren. Als Definition für eine entwicklungsbegleitende

Qualitätsbewertung elektronischer Systeme ist sie wenig hilfreich. Der Qualitätsbegriff des TQM erscheint auf dem ersten Blick ähnlich den klassischen Definitionen. Er ist jedoch weit umfassender, da deutlich mehr Einheiten (Prozesse, Mitarbeiter, etc.) im Fokus sind. Die Arbeit fokussiert sich jedoch auf das Produkt und betrachtet Qualität daher klassisch als Anforderungserfüllungsgrad. Nun ist es in der Produktentwicklung jedoch so, dass Anforderungen selten vollständig und korrekt sind. Ein Produkt kann die Anforderungen vollständig erfüllen und trotzdem ein Produkt schlechter Qualität sein. Wenn die Arbeit im Folgenden von Qualität spricht, ist folgendes darunter zu verstehen.

Definition Qualität: *„Die Qualität eines Produkts, bzw. eines seiner Teile ist der Grad der Erfüllung der an ihn gestellten Anforderungen und der Anforderungsqualität dieser Anforderungen selbst.“*

Die Begriffe Qualität und Produktqualität werden im weiteren Verlauf dieser Arbeit synonym verwendet.

1.2.2 Allgemeine Begriffe

Die Einführung der Begriffe erfolgt themenorientiert. Der Abschnitt beginnt mit einigen allgemeinen Begriffen der Produktentwicklung: Was zeichnet ein Projekt aus? Was ist ein „Produkt“? Was verbirgt sich hinter den Begriffen „Validierung“ und „Verifikation“, den typischen Aktivitäten der Qualitätssicherung?

- **Projekt:** Ein Projekt ist eine abgeschlossene, zeitlich begrenzte und zielgerichtete Kette von Aktionen (Büchel, 1987). Projekte werden als Rahmen für die Produktentwicklung aufgesetzt und enden mit der Fertigstellung der Entwicklung und dem Übergang des Produkts in die Fertigung.
- **Produkt:** Das Produkt ist das Ergebnis eines Entwicklungsprojekts. Es bezeichnet im Kontext dieser Arbeit ein Elektroniksystem, bzw. einen Teil dieses Systems, z.B. eine Hardware- oder Softwarekomponente, welche aus Sicht eines Zulieferers wiederum ein Produkt sein kann. Produkte haben in der Regel einen Lebenszyklus von der Produktidee bis zur Entsorgung (vgl. (Ehrlenspiel, 2007)), wobei sich die Arbeit auf den Teil der Produktentwicklung konzentriert. Die Arbeit verwendet die Begriffe Produkt und (Elektronik-)System synonym.
- **Validierung:** Validierung beantwortet die Frage „Baue ich das richtige Produkt?”. Die Arbeit unterscheidet zwischen Anforderungsvalidierung und Produktvalidierung:
    - Anforderungsvalidierung: Aktivitäten zum Nachweis, dass die definierten Anforderungen korrekt und vollständig sind.
    - Produktvalidierung: Aktivitäten zu Nachweis, dass das gelieferte (oder noch zu liefernde) Produkt seinen vorgesehenen Zweck erfüllt. (CMMI, 2010)

- **Verifikation:** Nachweis, dass die Arbeitsergebnisse die definierten Anforderungen umsetzt. Mit anderen Worten: Verifikation beantwortet die Frage „Baue ich das Produkt richtig?“. (CMMI, 2010)

Modelle und Metamodelle sind das in dieser Arbeit zentrale Werkzeug zur Beschreibung aller für eine Qualitätsbewertung notwendigen Daten. Die nächsten Punkte definieren den Begriff Metamodell sowie drei für diese Arbeit zentralen Modelltypen zur Repräsentation von Produkt- und Prozessdaten:

- **Produktmodell:** Der Begriff des Produktmodells stammt aus dem Bereich des Produktdatenmanagements. Produktmodelle bilden Produkte mit ihren für den gesamten Lebenszyklus relevanten Informationen digital ab (Eigner, 2009). Es kann aus mehreren Partialmodellen bestehen, die in der Regel mit verschiedenen Werkzeugen erzeugt werden. Im Gegensatz zum klassischen Begriff versteht diese Arbeit unter dem Produktmodell nicht die Menge aller produktrelevanten Dokumente, sondern die abstrakte integrierte Repräsentation des zu erstellenden Produktes inkl. seiner Zwischenergebnisse und seiner zugehörigen Analyse- und Verifikationsergebnisse. Dabei handelt es sich um eine bedarfsorientierte Repräsentation. Nur die für die entwicklungsbegleitende Qualitätsbewertung notwendigen Teile des Produktmodells werden abgebildet.
- **Partialmodell:** Ein Partialmodell beschreibt das Produkt unter einem bestimmten Blickwinkel. Dies ist beispielsweise eine Anforderungsdefinition, eine Systemarchitektur, der Entwurf einer Komponente oder die gesamte Implementierung einer Softwarekomponente (vgl. (Hausmann, 2009)). Bestandteile dieser Modelle können Analysen und deren Ergebnisse zugeordnet werden.
- **Design Artefakt:** Ein Design Artefakt ist ein Element eines Partialmodells. Beispiele für Design Artefakte sind Anforderungen, Komponenten und auch Analysen.
- **Prozessmodell:** Ein Prozessmodell beschreibt allgemein eine Menge an Prozessen, die aus einer Abfolge von Aktivitäten und Funktionen, ihrer In- und Outputs sowie involvierter Personen und Organisationen bestehen. Der Begriff kommt ursprünglich aus dem Bereich der Geschäftsprozessmodellierung. Diese Arbeit betrachtet dabei ausschließlich Modelle von Entwicklungsprozessen, keine weiteren Geschäftsprozesse des jeweiligen Unternehmens. Typische Prozessmodelle aus der Entwicklung sind das V-Modell XT, dass Wasserfallmodell oder der Rational Unified Process.
- **Metamodell:** Ein Metamodell ist ein explizites Modell und beschreibt Konstrukte und Regeln, die zur Erzeugung von Modellen einer bestimmten Domäne verwendet wird.[1] Ein Produktmetamodell beschreibt demnach die Bestandteile

[1] Basierend auf http://infogrid.org/wiki/Reference/PidcockArticle?story=20030115211223271

und Konzepte von Produktmodellen, ein Prozessmetamodell die Bausteine zur Definition konkreter Prozesse.

Von besonderer Relevanz für diese Arbeit ist der Begriff des Qualitätsmodells. Die Literatur verwendet diesen Begriff jedoch nicht immer eindeutig. Der Begriff des Qualitätsmodells wird sowohl für die Definition einzelner Eigenschaften als auch für ganze Frameworks verwendet, welches die Definition von spezialisierten Qualitätsmodellen ermöglicht (vgl. (Deißenböck, 2009), S. 34). Andersherum belegen verschiedene Ansätze identische Bestandteile eines Qualitätsmodells oft mit unterschiedlichen Begriffen. Die folgenden Definitionen schaffen bereits zu Beginn der Arbeit Klarheit:

- **Qualitätsmodell:** Ein Qualitätsmodell ist eine strukturelle Sammlung von Kriterien zur systematischen Qualitätsbewertung eines Gegenstands (vgl. (Deißenböck, 2009)).
- **Qualitätsmetamodell:** Entsprechend der gegebenen Metamodell Definition beschreibt ein Qualitätsmetamodell Konstrukte und Regeln zur Erzeugung spezieller Qualitätsmodelle (vgl. (Deißenböck, 2009)).
- **Qualitätsmodellierungsframework:** Ein Qualitätsmodellierungsframework umfasst Definition, Bewertung und Verbesserung von Qualität. Es beinhaltet üblicherweise ein Qualitätsmetamodell. Eine zugehörige Methodik beschreibt die Instanziierung konkreter Qualitätsmodelle und deren Verwendung zur Definition, Bewertung und Verbesserung von Qualität. (vgl. (Deißenböck, 2009))
- **Qualitätscharakteristik:** Ein Qualitätscharakteristik beschreibt eine Qualitätseigenschaft eines Gegenstands und ist Bestandteil eines Qualitätsmodells. Qualitätscharakteristika sind nicht zwangsweise messbar, können aber weiter bis hin zu Qualitätsattributen herunter gebrochen werden.
- **Qualitätsattribut:** Ein Qualitätsattribut ist eine spezielle Qualitätscharakteristik. Sie sind direkt mess- und nicht weiter verfeinerbar.

Nachdem der Abschnitt zentrale Begriffe dieser Arbeit eingeführt hat, schildert der folgende Abschnitt die Problemstellung, die mit einer entwicklungsbegleitenden Qualitätsbewertung einher gehen.

1.3 Problemstellung

Die Bewältigung der in Abschnitt 1.1 genannten Herausforderungen setzt eine Quantifizierung der geschaffenen Leistung, dem Output eines Produktentwicklungsprozesses voraus. Dieser Aussage bedarf es einer genaueren Erläuterung:

Im Hinblick auf die zweite Herausforderung ist Produktivität als eine dimensionslose Größe, festgelegt durch das Verhältnis von Output und Input (siehe z.B. (Pechlaner, 2005)). Ist die Bestimmung des Inputs als die Summe aller Investitionen in ein Projekt in Geldeinheiten verhältnismäßig einfach zu bestimmen, manifestiert sich der Output im erstellten Produkt und seiner Zwischenergebnisse. Die Bewertung des Outputs im Kon-

text der Produktentwicklung stellt jedoch eine besondere Herausforderung dar. Die hohe Anzahl abgebrochener oder zu teurer Entwicklungsprojekte zeigt, dass dies oftmals nicht gelingt, ansonsten würden Projektverantwortliche Probleme früher erkennen.

$$Produktivität = \frac{Komplexität \times Qualität}{Input\ (in\ Geldeinheiten)}$$

Formel 1: Entwicklungsproduktivität nach (Häusler, 2007)

Die Quantifizierung des Produktwerts (Häusler, 2007) fußt auf zwei Kennzahlen: die in der obigen Formel genannten Konzepte Komplexität und Qualität. Komplexität beschreibt dabei den technischen Wert des erstellten Produkts, welcher sich an quantitativen Eigenschaften des Produkts bzw. seines Modells messen lässt. Qualität ist entsprechend der in 1.2.1 gegebenen Definition der Grad der Anforderungserfüllung des Produkts bzw. einer seiner Komponenten, kombiniert mit dem Wert der Anforderungsqualität. Eine entwicklungsbegleitende Qualitätsbewertung liefert einen Einblick in den aktuellen Entwicklungsstand und stellt somit ein Beitrag für die Verbesserung des Projektcontrollings dar.

Eine Quantifizierung von Produktqualität elektronischer Systeme bereits während ihrer Entwicklung ist jedoch mit diversen Schwierigkeiten behaftet. Entwicklungsteams wenden eine Vielzahl an Methoden zum Nachweis von Qualitätseigenschaften an, welche jedoch jeweils immer nur einen Teilaspekt des Gesamtprodukts betrachten. Dies gilt bereits bei Entwicklungen, die sich auf eine Domäne konzentrieren, wie z.B. der Hardwareentwicklung. Hier existieren verschiedene EDA Tool spezifische Qualitätskriterien, wie z.B. Area oder Power, (Farrahi, 2000), Kriterien zur Bewertung von HDL (Hardware Description Language) Code (Mastretti, 1995), (Mastretti, 1996), (Protheroe, 2000), (Stefanoni, 1994), (Torroja, 1999) oder Arbeiten zur Bewertung von wieder verwendbaren Komponenten (Vörg, 2004), (Brand, 2004), in der Domäne als Intellectual Property (IP) bezeichnet. Darüber hinaus existieren im Bereich der Verifikation eine Vielzahl an Coverage Metriken, die eine Aussage über die Verifikationsabdeckung liefern. Analog verhält es sich in der Software- oder Systementwicklung, jede Domäne betrachtet für sich verschiedene Qualitätseigenschaften und verwendet diverse Methoden, welche diese absichern. Für eine Qualitätsbewertung reicht jedoch die punktuelle Betrachtung einzelner Qualitätskriterien nicht aus. Eine isolierte Betrachtung berücksichtigt keine gegenseitigen Abhängigkeiten zwischen einzelnen Teilaspekten. Wagner (Wagner, 2007a) fordert eine integrierte Auswertung dieser einzelnen Aspekte. Eine Aussage aus der Softwareentwicklung, die sich aber auf andere Domänen und domänenübergreifende Entwicklungsvorhaben übertragen lässt.

Dabei beschäftigen sich Forschung und Praxis mit diesem Problem bereits seit Jahrzehnten. So existieren vor allem in der Softwareentwicklung Bestrebungen zur Definition allgemeingültiger Softwarequalitätsmodelle. Bereits Ende der 70 Jahre des letzten Jahrtausends entstanden die klassischen Ansätze von Boehm (Boehm, 1978) und Bowen (Bowen, 1976), in den späten 80ern von Deutsch (Deutsch, 1988) und Kitchenham

(Kitchenham, 1987), sowie Nance (Nance, 1992) Anfang der 90er Jahre, um nur einige zu nennen. Für eine detaillierte Untersuchung der Vor- und Nachteile dieser und weiterer Ansätze sei auf Abschnitt 2.3 verwiesen. Grundsätzlich lässt sich jedoch schon jetzt sagen: Viele Ansätze sind, wie noch gezeigt wird, zu abstrakt oder scheitern an der Übertragbarkeit auf einen konkreten Unternehmens- bzw. Projektkontext. Dies verwundert bei näherer Betrachtung existierender Qualitätsdefinitionen als Grad der Anforderungserfüllung (siehe 1.2.1) nicht. Jedes Entwicklungsprojekt ist einzigartig und stellt unterschiedliche Anforderungen an das zu entwickelnde Produkt. In einem statischen Qualitätsmodell finden sich unter diesen Umständen immer ungültige, falsch gewichtete oder fehlende Qualitätscharakteristika. Beschreibungen einzelner Qualitätscharakteristika, wie z.B. Wiederverwendbarkeit, lassen sich zwar theoretisch in allen Projekten anwenden. Ist Wiederverwendbarkeit jedoch nur eine untergeordnete Anforderung, ist eine Bewertung dieser Eigenschaft nicht oder nur mit einer entsprechenden Gewichtung in eine Qualitätsbewertung mit einzubeziehen.

Eine weitere zentrale Herausforderung einer zielorientierten Qualitätsbewertung ist die Berücksichtigung des Entwicklungsprozesskontexts. Deutlich wird dies am Bespiel sicherheitskritischer Systeme. Eine hohe Kritikalität eines Systems resultiert in deutlich mehr Validierungs- und Verifikationsaktivitäten im Entwicklungsprozess. Die zu verifizierenden Anforderungen sind im Wesentlichen dieselben. Es sind die angewandten Methoden, die den Unterschied ausmachen und die Qualitätsziele eines Projekts mit beeinflussen. Will ein Projektleiter die Ergebnisse der Qualitätsbewertung für die Projektsteuerung verwenden, reicht in diesem Fall eine bloße Betrachtung der Anforderungserfüllung nicht aus. Es ist eine kontextspezifische Bewertung durchzuführen. Habe ich die Anforderungserfüllung mit den notwendigen Mitteln nachgewiesen? Auch der Ablauf eines Projekts, also die Projektplanung hat einen Einfluss auf eine zielorientierte Qualitätsbewertung. Je nach Entwicklungsphase (z.B. Anforderungsdefinition oder Architekturentwurf) gelten z.B. unterschiedliche Qualitätsziele, die nur eine Teilmenge der Qualitätsdefinition betrachten. Je nach Vorgehen existieren (Zwischen-) Meilensteine, welche lediglich die Erfüllung bestimmter Zwischenziele fordern. Beispiel: Drei Monate vor Release sollte ein Anforderungserfüllungsgrad aller kritischen Anforderungen von 90% vorliegen, um das Risiko zu minimieren den letzten Meilenstein nicht einzuhalten.

Die Definition eines allgemeingültigen bzw. statischen Qualitätsmodells zur Überwachung von Produktqualität während der Elektronikentwicklung ist daher nicht realistisch. Es ist eine flexible Bewertung von Produktqualität auf Basis formulierter Anforderungen und unter Berücksichtigung des Prozesskontexts erforderlich. Vor allem ist eine Integration der Prozess- und Produktdaten mittels einer gemeinsamen Modellierungsgrundlage notwendig, um eine flexible Qualitätsbewertung auch zu automatisieren. Wie die Untersuchung des Stands der Technik zeigen wird, existiert kein Ansatz für eine entwicklungsbegleitende Produktqualitätsüberwachung im Allgemeinen noch für die Elektronikentwicklung im Speziellen, welches alle genannten Probleme adressiert.

1.4 Zieldefinition und Beitrag der Arbeit

Die Arbeit adressiert die genannten Probleme des vorherigen Absatzes und entwickelt einen Ansatz, der die Qualität elektronischer Systeme prozessorientiert definiert, die Definition auf einzelne Projekte kalibriert und diese regelmäßig auf Basis aktueller Entwicklungsdaten auswertet. Die weiteren Abschnitte formulieren die Zielkriterien an die zu entwickelnde Lösung und den Beitrag der Dissertation.

1.4.1 Zieldefinition

Aus der Problemstellung (Abschnitt 1.3) lassen sich Ziele an eine entwicklungsbegleitende Qualitätsbewertung für die Elektronikentwicklung ableiten. Die Auflistung verfolgt folgenden Zweck: zum einen lassen sich an den Zielen vorhandene Ansätze aus Literatur und Praxis auf ihre Tauglichkeit hin überprüfen. Zum anderen ermöglichen sie die Verifikation der im Rahmen dieser Arbeit erstellten Konzeption. Quellen sind sowohl wissenschaftliche Veröffentlichungen als auch definierte Ziele und Anforderungen von Industriepartnern des ARTEMIS Projekts CESAR[2] der Anwendungsdomänen Automotive (CESAR, 2010a), Aerospace (CESAR, 2010b), Automatisierung und Bahntechnik (CESAR, 2010c):

Ziel 1 – Einheitliche Qualitätsdefinition als Grundlage einer gemeinsamen Betrachtung unterschiedlicher Qualitätscharakteristika. In jeder Domäne (Software-, Hardware-, Systementwicklung) gibt es eine Vielzahl an Qualitätseigenschaften, die jedoch in der Regel einzeln analysiert werden. Farrahi (Farrahi, 2007) nennt z.B. verschiedene Qualitätscharakteristika für Hardware Komponenten, die nur gemeinsam interpretierbar sind. Um diese einzelnen Bewertungen gemeinsam zu interpretieren, ist eine einheitliche Qualitätsdefinition notwendig. Wagner et al. (Broy, 2005), (Wagner, 2007a) beschreiben eine ähnliche Anforderung bezüglich einer ganzheitlichen Qualitätsbewertung von Software Systemen.

Ziel 2 – Effiziente Kalibrierung auf einzelne Projekte. Neben einer einheitlichen Qualitätsdefinition ist die Anpassbarkeit dieser auf einen spezifischen Projektkontext zu unterstützen, dabei aber gleichzeitig die Möglichkeit der Wiederverwendung einer Qualitätsdefinition über mehrere Projekte hinweg nicht zu vernachlässigen. Diese Herausforderung ist ebenfalls Bestandteil einer Forschungsagenda für Software Qualität (Wagner, 2010), lässt sich aber auch auf andere Ingenieursdisziplinen übertragen.

Ziel 3 – Integration von Verifikations- & Validierungsergebnissen. In der Elektronikentwicklung wird eine Vielzahl an Verifikations- & Validierungsaktivitäten zum Nachweis bestimmter Produkteigenschaften durchgeführt. Diese liefern die notwendigen Informationen zur Bewertung des aktuellen Qualitätsstands. Verifikations- und Validierungsergebnisse sind zusammenzuführen, um eine ganzheitliche Qualitätsbewertung entsprechend der Qualitätsdefinition und projektspezifischer Anforderungen zu

[2] www.cesarproject.eu

ermöglichen. Die Notwendigkeit der Integration von V&V Ergebnissen wurde auch auf einer EDA Konferenz (Bacchini, 2007) diskutiert.

Ziel 4 – Domänenübergreifende Qualitätsbetrachtung. Die Entwicklung heutiger komplexer Elektroniksysteme ist im Regelfall in mehrere Architekturen mit unterschiedlichen Arten von Komponenten (z.B. Hardware, Software) und jeweils mehreren Zwischenergebnissen unterteilt. Eine Lösung für das Qualitätscontrolling in der Elektronikentwicklung muss diese verschiedenen Arten an Komponenten und deren Zwischenergebnisse berücksichtigen, um eine Übersicht des Gesamtprojekts zu schaffen. Die Notwendigkeit der Qualitätsbewertung auf verschiedenen Ebenen nennt ebenfalls (Deißenböck, 2008).

Ziel 5 – Entwicklungsbegleitende Auswertung von Produktqualität. Die Bewertungen sind kontinuierlich und parallel zu den eigentlichen Entwicklungsaktivitäten durchzuführen, um bereits vor Meilensteinterminen eine Aussage über den aktuellen Meilensteinerfüllungsgrad, sowie über die Einhaltung des Termins zu treffen. Die Entwicklung über die Zeit offenbart Informationen, die einzelne Snapshots nicht liefern (Deißenböck, 2008). Basierend auf der Betrachtung der zeitlichen Entwicklung von Informationen wie z.B. die Anzahl offener Probleme oder der aktuellen Testabdeckung lassen sich verschiedene Rückschlüsse auf den Projektstatus ziehen.

Ziel 6 – Berücksichtigung von Prozess- und Projektkontext. Um durchgeführte Qualitätsbewertungen für die Prozesskontrolle und –steuerung zu operationalisieren, ist ebenfalls Prozess- und Projektkontext zu berücksichtigen. Die durchgeführten Bewertungen auf Produktebene sind mit definierten Prozess- und Projektzielen abzugleichen. Neben Hauptmeilensteinen existieren in einer typischen Projektdefinition mehrere Quality Gates, an denen z.B. ein bestimmter Stand der Verifikation erlangt sein muss (als Beispiel sei hier die maximale Anzahl offener Fehler genannt), um zu garantieren, dass das Projekt im Plan bleibt. In dieser Form werden Quality Gates als zusätzliche Kontrollpunkte verwendet. Dementsprechend ist es zu ermöglichen Design Artefakte (seien es Anforderungen, Architekturen oder einzelne Komponenten) an mehreren Zeitpunkten gegen jeweils unterschiedliche Zielwerte zu bewerten. Darüber hinaus existieren z.B. in sicherheitskritischen Domänen Prozessanforderungen durch Standards, die je nach Kritikalität des entwickelten Systems bestimmte Verifikationsmethoden vorschreiben. Für die Qualitätskontrolle sind also nicht nur die Testergebnisse selbst relevant, sondern auch die Art der Methode, wie der Test durchgeführt wurde.

Ziel 7 – Automatisierung von Datenerfassung, –integration und -analyse. Die Bewertungen sind auf Grund der anzunehmenden Fülle an messbaren Parametern und notwendigen Aggregationen soweit es geht zu automatisieren, um die Entwickler nicht bei der Arbeit zu behindern. Bereits in einer Studie für Software Project Control Centers (Ciolkowski, 2007) wurde dies als ein wichtiger Punkt genannt. Betrachtet man mehr als nur die reine Softwareentwicklung, existieren noch mehr Zwischenergebnisse und Analyseergebnisse. Ziel 5 macht darüber hinaus eine regelmäßige Erfassung des aktuellen Entwicklungsstands (relevante Bestandteile von Produktmodellen und Analyseer-

gebnisse) notwendig. Dabei ist zu berücksichtigen, dass die notwendigen Daten zur Qualitätsbewertung in der Regel über mehrere Produktmodelle und Datenquellen verteilt vorliegen.

Ziel 8 – Anpassbarkeit an bestehende Entwicklungsprozesse. Da sich Unternehmen in den seltensten Fällen gleichen und Entwicklungsprozesse innerhalb eines Unternehmens je nach Projekt unterschiedlich ausgeprägt sind, muss die Lösung Methoden für die Anpassung auf spezifische Entwicklungsprozesse (z.B. nur Architekturentwicklung oder nur Hardwareentwicklung) vorsehen. Dabei ist die Lösung keinesfalls Ersatz bestehender Methoden und Prozesse, noch darf die Lösung auf diese beschränkend wirken.

1.4.2 Beitrag

Der Beitrag dieser Arbeit lässt sich mit der folgenden Aussage zur wissenschaftlichen Wertschöpfung zusammenfassen als:

„Erstellung eines Konzepts zur entwicklungsbegleitenden Produktqualitätsbewertung in der Elektronikentwicklung basierend auf einer prozessorientierten Qualitätsdefinition und einer einheitlichen Repräsentation qualitätsrelevanter Produktdaten unter Verwendung integrierter Produkt- und Prozessmetamodelle.“

In diesem Zusammenhang erbringt die vorliegende Arbeit folgende Beiträge:

1. **Entwicklung eines Qualitätsmodellierungsframeworks zur prozessorientierten Qualitätsdefinition und -bewertung.** Die Arbeit entwickelt eine Lösung, welche die Ergebnisse durchgeführter Maßnahmen zur Qualitätssicherung in der Elektronikentwicklung zusammenführt, entsprechend des Prozess- und Projektkontexts interpretiert und für Prozesskontrolle und -steuerung operationalisiert. Hierfür ist die Lösung mit dem Anforderungsmanagement eines Projekts zu verknüpfen. Ziel ist eine integrierte Qualitätsbewertung, welche definierte (funktionaler und nicht funktionaler) Anforderungen unddurchgeführte Verifikations- & Validierungsaktivitäten berücksichtigt. Um diese produktorientierten Qualitätsinformationen im Kontext ihres Entwicklungsprozesses zu bewerten und für Kontrolle und Steuerung ebendiesen zu nutzen, ist innerhalb des Qualitätsmodellierungsframeworks die Produktsicht mit einer Prozesssicht inkl. zugehöriger Qualitätsziele geeignet zu verknüpfen.
2. **Definition eines Vorgehensmodells.** Eine umfassende Lösung erfordert die Definition eines Vorgehensmodells basierend auf dem entwickelten Framework, da Projekte und damit auch konkrete Instanzen des Qualitätsmodellierungsframeworks immer unternehmensspezifisch sind. Das Vorgehensmodell beschreibt die Schritte zur Kalibrierung und Anwendung des Frameworks in einer speziellen Anwendungsdomäne und Unternehmen und deren Anbindung an die dort existierenden Entwicklungsprozesse, -methoden und –werkzeuge.

3. **Implementierung:** Das Qualitätsbewertungsframework wird implementiert und die Konzeption des Qualitätsmodellierungsframeworks anhand von zwei Anwendungsbeispielen aus der Elektronikentwicklung evaluiert.

1.5 Struktur der Arbeit

Die vorliegende Arbeit gliedert sich in sechs Kapitel und folgt dem typischen ingenieurmäßigen Vorgehen aus Problemstellung, Zieldefinition, Stand der Technik und der Identifikation des Handlungsbedarfs, Konzeption, Implementierung und Evaluierung des entwickelten Konzepts.

Problemstellung und Zielsetzung behandeln die Abschnitte im ersten Kapitel dieser Arbeit. Diese legen Wert auf die umfassende Definition der, an die zu konzipierende Lösung gestellten Ziele. Die Analyse des Stands der Technik wird entsprechend der definierten Ziele aus diesem Kapitel durchgeführt. Auch die eigene Lösung zur Adressierung der Problemstellung muss sich an ihnen messen. Der Stand der Technik wird in Kapitel 2 abgehandelt. Hier treten aktuelle Methoden für das Projektcontrolling, des quantitativen Messens und des Qualitätsmanagements in den Vordergrund. Weiterhin untersucht der Stand der Technik Ansätze des Anforderungsmanagements und der Qualitätsmodellierung und –bewertung. Die Arbeit konzentriert sich dabei auf Ansätze aus den Bereichen des Software und Systems Engineerings, den relevanten Bereichen der Elektronikentwicklung. Die Abschnitte dieses Kapitels untersuchen die Eignung existierender Ansätze, bezugnehmend auf die Ziele dieser Arbeit und identifiziert Schwachstellen und den Handlungsbedarf. Damit schaffen die ersten zwei Kapitel die Grundlage für die Konzeption einer angemessenen Strategie zur Lösung der Problemstellung.

Kapitel 3 detailliert das entwickelte Lösungskonzept. Basierend auf dem identifizierten Handlungsbedarf werden Anforderungen an den Lösungsansatz abgeleitet und durch ein entwickeltes Qualitätsmodellierungsframework inkl. Vorgehensmodell zur prozessorientierten Qualitätsdefinition und –bewertung für die Elektronikentwicklung umgesetzt. Darauf aufbauend beschreibt Kapitel 4 die Umsetzung des Konzepts. Kapitel 5 nutzt den Prototyp und demonstriert dessen Anwendung anhand von zwei Anwendungsbeispielen aus der Elektronikentwicklung. Kapitel 6 schließt letztlich mit einer Zusammenfassung dieser Arbeit und einem Ausblick auf mögliche Erweiterungen.

2 Stand der Technik

Dieses Kapitel beschreibt den Stand der Technik in den drei Themengebieten, die dieser Arbeit am nächsten stehen und bewertet existierende Methoden und Werkzeuge anhand der Ziele dieser Arbeit (siehe Abschnitt 1.4.1). Basierend auf dieser Analyse wird der weitere Handlungsbedarf identifiziert, um die motivierten Ziele zu erfüllen. Das Ergebnis dieser Untersuchung ist Grundlage für die Entwicklung des Lösungskonzepts in Kapitel 3. Folgende drei Gebiete stehen dieser Arbeit am nächsten:

- **Projekt- und Qualitätscontrolling in der Produktentwicklung:** Die vorliegende Arbeit entwickelt ein prozessorientiertes Produktqualitätsmonitoring für die Elektronikentwicklung, um Prozesskontrolle und –steuerung in Projekten durch die Bestimmung des aktuellen Qualitätsstands zu unterstützen. Abschnitt 2.1 untersucht daher existierende Ansätze aus dem Bereich des Projektcontrollings im Allgemeinen und dem Qualitätscontrolling im Speziellen. Darüber hinaus untersucht dieser Absatz Ansätze des quantitativen Messens, die als Hilfestellung zum Aufsetzen quantitativer Messprogramme für Projekt und Qualitätscontrolling anwendbar sind.
- **Requirements Engineering:** Entsprechend der Qualitätsdefinition aus Abschnitt 1.2.1, sind Anforderungen elementarer Bestandteil einer zielorientierten Qualitätsbewertung. In der Elektronikentwicklung ist es industrielle Praxis basierend auf definierten Anforderungen und Validierungs- und Verifikationsergebnissen eine Aussage über die Produktqualität zu treffen (siehe (CESAR, 2010e)). Das zweite untersuchte Themengebiet umfasst daher den Bereich des Requirements Engineerings. Dieses Themengebiet ist – wie Abschnitt 2.2 zeigen wird - untrennbar mit dem Thema Qualitätssicherung verknüpft. Der Abschnitt untersucht speziell Ansätze zur Anforderungsmodellierung und -verfolgung.
- **Qualitätssicherung in der Produktentwicklung:** Qualitätssicherung von Produkten ist eines der elementaren Aufgaben in der Elektronikentwicklung, in der es um den Nachweis geforderter Produkteigenschaften geht. Es liegt daher nahe diesen Bereich in einer Arbeit zur entwicklungsbegleitenden Bestimmung von Produktqualität genauer zu untersuchen. Abschnitt 2.3 führt in die Aktivitäten rund um den Bereich der Verifikation & Validierung im Software und Systems Engineering ein und widmet sich darauf dem Themengebiet der Qualitätsmodellierung und –bewertung. Dieser Bereich steht dieser Arbeit offensichtlich und unmittelbar am nächsten und umfasst zu einem nicht unerheblichen Teil Ansätze aus der Softwareentwicklung.

Der letzte Abschnitt fasst alle identifizierten Schwachpunkte im Stand der Technik zusammen und definiert den Handlungsbedarf für diese Arbeit auf dem das Konzeptkapitel 3 aufbaut.

2.1 Projekt- und Qualitätscontrolling in der Produktentwicklung

Die Unterstützung des Projektcontrollings in der Elektronikentwicklung durch eine begleitende Auswertung von Produktqualität steht im Fokus der Dissertation, da in Anbetracht der vielen verzögerten, zu teuren oder völlig gescheiterten Projekte besonderer Handlungsbedarf besteht. Dieser Abschnitt untersucht Ansätze aus Forschung und Praxis des Projekt- und Qualitätscontrollings sowie des quantitativen Messens.

2.1.1 Projektcontrolling

Horvath (Horvath, 2006) definiert Controlling als „Subsystem der Führung, das Planung und Kontrolle sowie Informationsversorgung systembildend und systemkoppelnd ergebniszielorientiert koordiniert und so die Adaption und Koordination des Gesamtsystems unterstützt.“ Nach DIN 69904 ist Projektcontrolling als "Prozesse und Regeln, die innerhalb des Projektmanagements zur Sicherung des Erreichens der Projektziele beitragen" definiert. Diese Prozesse reichen von der Datenerfassung, über den Vergleich mit den Plandaten und der Feststellung von Abweichungen, bis hin zur Bewertung der Konsequenzen und dem Mitwirken bei der Planung von Gegenmaßnahmen und ihrer Überwachung (vgl. (Projektmagazin, 2011)). Diese Definition deckt sich mit der Beschreibung des PMBOK: „Control“ als Technik, zu der Soll-Ist-Vergleich, Trendanalysen, Abweichungsanalysen, die Bewertung möglicher Handlungsalternativen und die Empfehlung geeigneter Maßnahmen zur Steuerung des Projekts zählen“ (PMI, 2008).

Klassische Methoden des Projektcontrollings

Im Zentrum der Projektüberwachung steht der Soll-Ist-Vergleich im Hinblick auf Termin-, Kosten- oder Qualitätsziele. Wichtigste und gleichzeitig schwierigste Aufgabe der Projektüberwachung ist die Kontrolle der Qualität bzw. des Sachfortschritts, welche den Schwerpunkt dieser Arbeit darstellt. Grundlage sind definierte Soll-Werte und deren Vergleich mit dem aktuellen Ist-Stand. Eine bewährte Methodik ist in diesem Zusammenhang die Earned Value Analyse (Walenta, 2001). Die Earned Value Analysis basiert auf dem allgemeinen Ansatz, jedes Arbeitspaket über die geplanten Kosten zu bewerten. Diese Zahl wird nach Abschluss und Freigabe des Arbeitspakets dem Earned Value, sinngemäß also dem bereits erreichten Produktwert, zugerechnet. Die Bestimmung des Earned Value zu einem Zeitpunkt ermöglicht so die Bewertung der geleisteten Arbeit in Relation zur Zeit und den bis dahin tatsächlich angefallenen Kosten. An dieser Stelle ist auch die Verwendung anderer Metriken zur Bestimmung des Arbeitsfortschritts denkbar, jedoch bei der Earned Value Analyse nicht weiter ausgeführt. Für eine detaillierte Beschreibung sei auf (Walenta, 2001) verwiesen.

Abweichungen des Ist vom Soll-Wert führen zu Korrekturmaßnahmen oder Plananpassung (Litke, 1995). Zur gezielten Überwachung von Kosten und Terminen im Projekt-

verlauf eignen sich Meilensteintrend- und Kostenverlaufsdiagramme (Wedelstädt, 2001). Meilensteintrenddiagramme bieten eine anschauliche Darstellung des Projektverlaufs anhand der geplanten Meilensteintermine und dienen der Indikation und Prognose möglicher Planabweichungen. Ein ebenfalls viel genutztes Instrument sind Quality Gates, welche festgelegte Synchronisationspunkte innerhalb eines Entwicklungsvorhabens darstellen. Ein Quality Gate ist ein ergebnisorientierter Zeitpunkt, der durch produkt- bzw. prozessspezifische Leistungen definiert wird (Wildemann, 2003). Ein Gate ist erst passierbar, wenn das entwickelte Ergebnis und das weitere Vorgehen freigegeben sind. Grundlage der Freigabe sind zwischen Kunden und Auftragnehmer vereinbarte oder aber als Best Practice bekannte Ziele. Auch hier fußt die Berechnung der Zielerreichung auf einem Soll-Ist Abgleich definierter Messkriterien.

Die Bestimmung des Ist-Stands nimmt offensichtlich eine zentrale Rolle bei all diesen Methoden ein. Häufig fordert ein Projektleiter Informationen von den jeweiligen Verantwortlichen an und bestimmt den Fortschritt basierend auf diesen subjektiven Aussagen (Sieg, 2007). Metriken und Indikatoren sind ein Mittel, um diese Bewertungen zu optimieren. An verschiedensten Stellen wird auf Metriken und Indikatoren, wie z. B. Lines of Code (LOC), Anzahl abgeschlossener Arbeitspakete oder erfolgreiche Reviews als Ersatzgrößen zurückgegriffen. Nachteil dieser Indikatoren ist, dass sie oft nur einen indirekten Bezug zum tatsächlichen Projektstatus haben und deshalb nur eingeschränkt eine Aussage auf den Sachfortschritt zulassen. Elementar ist ein erfahrener Projektleiter, der die richtigen Metriken auswählen und interpretieren muss. Ohne die Sachkenntnis eines erfahrenen Projektleiters ist eine Projektbewertung daher zwangsläufig ungenau und mit Unsicherheit behaftet.

Die klassischen Methoden des Projektcontrollings können keine ausreichende Antwort auf die Frage einer objektiven Bestimmung des Ist-Stands, speziell der Qualität in einem Projekt liefern. In der industriellen Praxis entwickeln viele Unternehmen ihre eigenen Anwendungen mit Hilfe von Business Intelligence Lösungen zur systematischen Analyse (Sammlung, Auswertung und Darstellung) von Daten. Diese Lösungen sind jedoch spezifisch für das jeweilige Unternehmen, mitunter sogar für bestimmte Bereiche und Projekte. Diese verschiedenen Dashboard basierten Ansätze werden an dieser Stelle nicht weiter erläutert, es sei jedoch angemerkt, dass sie natürlich für die Umsetzung einer Qualitätsauswertung verwendbar sind, jedoch keinerlei methodische Unterstützung bei der Umsetzung liefern.

Software Project Control Center

„Software Projekt Control Center“ (SPCC) sind ein im Software Engineering entstandenes Konzept für das Software Projektcontrolling. Sie stellen konkrete Werkzeugunterstützung dar und gehen damit weiter als die bisher vorgestellten Arbeiten. Ein SPCC ist gemäß Münch (Münch, 2004) als ein Mittel zur prozessbegleitenden Interpretation und Visualisierung von Messdaten definiert. SPCC sollen mit Hilfe von Messungen und expliziten Modellen (z.B. Prozessmodelle, Produktmodelle oder Qualitätsmodelle) dabei helfen, Softwareentwicklungsprojekte besser zu kontrollieren, deren Leistung zu

bestimmen und systematische Qualitätssicherung durchzuführen (Heidrich, 2006). Das am IESE in Kooperation mit der Universität Kaiserslautern entwickelte SPCC nennt sich G-SPCC und steht für einen zielorientierten (engl.: goal-oriented) SPCC Ansatz.

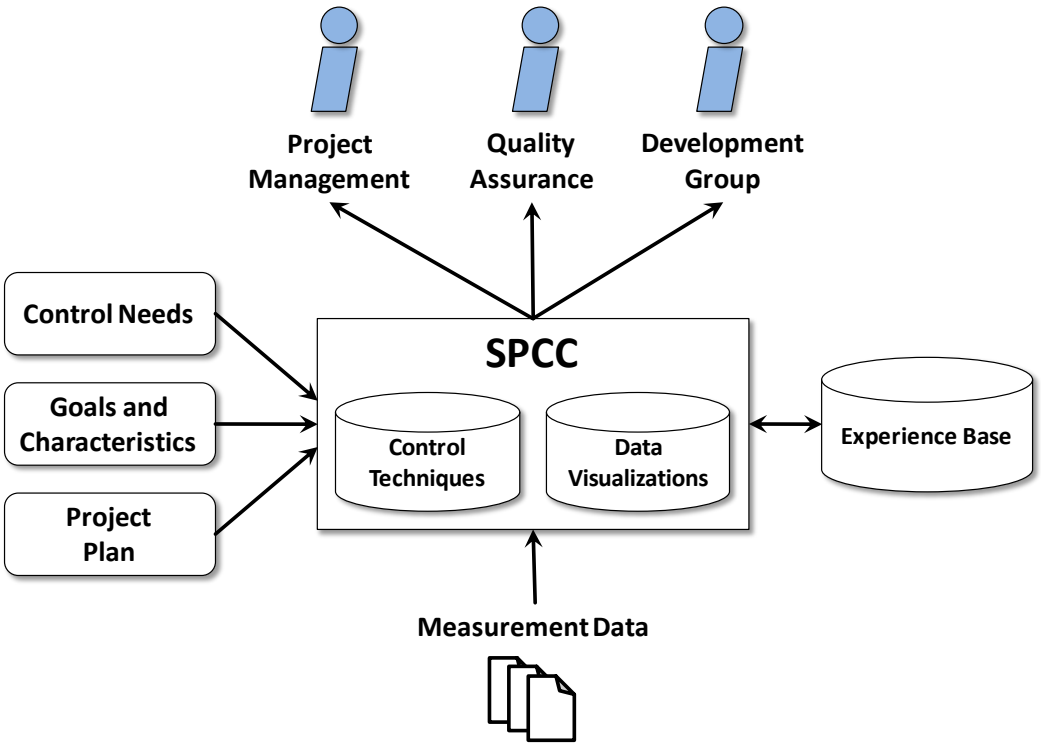

Abbildung 3: G-SPCC Framework nach (Heidrich, 2006)

Abbildung 3 zeigt das G-SPCC Framework mit seinen Ein- und Ausgaben. Daten werden während des Projekts erfasst und gemäß der Ziele und Charakteristika, sowie Informationen über den Plan des betrachteten Projekts und sogenannten „Control Needs“ ausgewertet. Control Needs werden dabei gemäß Burghardt (Burghardt, 2002) in die vier Bereiche Zeit- und Planeinhaltung, Kosten- und Aufwandseinhaltung, Projektfortschrittsüberwachung und Qualitätskontrolle unterschieden. Der Kern des SPCC besteht aus Techniken zur Interpretation der Daten, sowie die Visualisierung der Ergebnisse, bei der die Rolle des Benutzers (z.B. Projektmanager, Qualitätsmanager oder Entwickler) mit berücksichtigt wird. Der Prozess zur Interpretation und Visualisierung wird durch eine „Experience Base“ unterstützt, welche Erfahrungen vergangener Projekte enthält. Das Framework deckt dabei durch die Definition sogenannter Visualisation Catenas (VC) (Heidrich, 2003) die Relationen zwischen erfassten Daten, deren Analyse und der zugehörigen Visualisierung ab. VCs können entsprechend von Unternehmensanforderungen dynamisch hinzugefügt und angepasst werden.

Der Ansatz des IESE stellt im Umfeld der Software Project Control Center eines der aktuellsten und umfangreichsten Ansätze dar. Für eine tiefere Untersuchung des Stands der Technik speziell in diesem Bereich sei auf (Münch, 2004) verwiesen. Die Arbeiten beinhalten jedoch kein systematisches Vorgehen für eine entwicklungsbegleitende Qua-

litätsüberwachung, wie es in dieser Arbeit angestrebt wird. Die Evaluierung innerhalb des BMBF geförderten Forschungsprojekts Soft-Pit[3] verdeutlicht dies. Das G-SPCC diente in dem geschilderten Anwendungsbeispiel (Ciolkowski, 2007) als Implementierungsbasis. Probleme waren unter anderem der manuelle zeitlich aufwändige Datenimport. Insbesondere beinhalten Qualitätsbetrachtungen lediglich die Codequalität, ein typisches Charakteristikum der Softwareentwicklungsdomäne. Die Elektronikentwicklung produziert jedoch weit mehr als nur Software Code, und es ist mehr als nur dessen Wartbarkeit während der Entwicklung von Belang. Die Ziele 1 bis 4 sind daher durch SPCC nicht erfüllt. Sie stellen jedoch teilweise eine Lösung von **Ziel 5 - Entwicklungsbegleitende Auswertung** und **Ziel 7 - Automatische Datenerfassung, -integration und –auswertung** dar. Die automatische Erfassung und Auswertung ist jedoch nicht auf die Qualitätsbewertung spezialisiert und reicht daher nicht vollständig aus. **Ziel 8 - Einbettung in existierende Entwicklungsprozesse** wird erfüllt, da SPCC für verschiedene existierende Entwicklungsprozesse einsetzbar sind. Alleinstehend hat dies aber für diese Arbeit keinen großen Wert.

2.1.2 Qualitätscontrolling

Neben den beschriebenen Projektcontrolling Ansätzen, existieren Methoden des Qualitätsmanagements, die sich auf verschiedene Managementbereiche rund um das Thema Qualität inkl. des Qualitätscontrollings spezialisiert haben. Qualitätsmanagement ist ein die Entwicklung unterstützender Prozess, der alle „aufeinander abgestimmte Tätigkeiten zum Leiten und Lenken einer Organisation bezüglich Qualität“ (vgl. (Zollondz, 2006, S. 210)) umfasst.

Begriffe und Definitionen der Aktivitäten des Qualitätsmanagements unterscheiden sich je nach Branche und Autor teilweise erheblich. Diese Arbeit folgt der Unterteilung in fünf Teilaktivitäten nach Deißenböck (Deißenböck, 2009), die wiederum auf existierenden Standards und Arbeiten von Juran (Juran, 1988), Crosby (Crosby, 1995) und Deming (Deming, 2000) basiert. Die für diese Arbeit grundlegenden Definitionen der fünf Teilaktivitäten des Qualitätsmanagements sind wie folgt:

- *Qualitätsplanung:* Die Qualitätsplanung legt die Qualitätsziele für den Entwicklungsprozess fest. Darüber hinaus definiert die Planung den Qualitätssicherungsprozess, der zur Qualitätskontrolle notwendig ist.
- *Qualitätsbewertung:* Die Qualitätsbewertung vergleicht Qualitätscharakteristika mit Qualitätszielen, um deren Erfüllung zu bewerten.
- *Qualitätskontrolle:* Die Qualitätskontrolle beinhaltet die Qualitätsbewertung und umfasst darüber hinaus die Initiierung notwendiger korrektiver Maßnahmen bei Problemen.

[3] http://www.soft-pit.de

- *Qualitätssicherung:* Die Qualitätssicherung beinhaltet die Qualitätskontrolle, geht jedoch darüber hinaus, in dem es auch die Mittel zur Verfügung stellt, die für eine effiziente Qualitätskontrolle notwendig sind, z.B. die Auswahl geeigneter Methoden und Werkzeuge.
- *Qualitätsverbesserung:* Die Qualitätsverbesserung fokussiert die Verbesserung der Qualitätsplanung und –sicherungsprozesse selbst, nicht jedoch des Produkts. Es versucht die Wahrscheinlichkeit zukünftiger Qualitätsprobleme und –kosten zu minimieren.

Überträgt man die Projektcontrolling Definition nach Horvath auf das Qualitätscontrolling, besteht dieses aus der Qualitätsplanung –bewertung und –kontrolle, welche die Informationsversorgung umfasst, um die Koordination und Adaption des Gesamtsystems mit einem Fokus auf Qualität zu unterstützen.

Die vorliegende Arbeit liefert einen Beitrag für den Bereich der Qualitätsplanung, in dem es einen einheitlichen Ansatz zur Qualitätsdefinition und Qualitätszielen liefert. Ebenfalls unterstützt sie die Qualitätsbewertung durch die entwicklungsbegleitende Überwachung der Qualitätsziele und stellt die Informationen für die Qualitätskontrolle zur Verfügung. Dieser Querschnitt durch die drei Teilprozesse des Qualitätsmanagements wird im weiteren Verlauf dieser Arbeit als Produktqualitätsmonitoring bezeichnet. Der weitere Abschnitt untersucht die bekanntesten Methoden des Qualitätsmanagements mit Fokus auf den Bereichen des Produktqualitätsmonitorings.

Deming Zyklus/ Plan-Do-Check-Act

Der Deming Zyklus, auch Plan-Do-Check-Act oder PDCA-Zyklus genannt, geht auf den Statistiker Shewhart zurück, wurde aber von Deming in den 50er Jahren formuliert. Der Deming Zyklus ist heutzutage das grundlegende Handlungsschema jedweder Verbesserungsaktivitäten (Zollondz, 2006). Es folgt eine kurze Erläuterung der vier Phasen (vgl. (Zollondz, 2006)):

- *Planungsphase (plan)*: Analyse der Ist-Situation auf Grundlage der zu ermittelnden Daten. Planung der Datenerhebung, -analyse und –auswertung. Prüfpunkte festlegen. Verbesserungsplan entwerfen.
- *Planungsumsetzungsphase (do):* Schulung der Mitarbeiter, Verbesserungen umsetzen.
- *Prüfphase (check):* Daten ermitteln und mit Zielsetzung der Planungsphase abgleichen.
- *Aktionsphase (act):* Bei Zielerfüllung Ergebnis standardisieren und unternehmensweit einführen. Bei Abweichungen solange iterieren, bis Übereinstimmung besteht, bzw. ab einer bestimmten Iterationsanzahl abbrechen.

Dieses grundlegende Vorgehen findet sich auch in den im Folgenden vorgestellten Methoden und Standards wieder. Es ist jedoch zu Allgemein, um die Ziele dieser Arbeit zu adressieren. Werkzeuge werden offen gelassen.

Six Sigma

Six Sigma ist sowohl statistisches Qualitätsziel als auch Management Instrument, welches vor allem darauf abzielt Prozesse bis zur Null-Fehler-Qualität zu optimieren. Sie wurde in den 80er Jahren von Motorola als Verfahren zur Qualitätskontrolle entwickelt und erlangte in den 90er Jahren durch die Anwendung bei General Electric größere Verbreitung. Die folgende Beschreibung basiert auf Toutenburg (Toutenburg, 2009).

Die klassische Six Sigma Methodik, die primär in der Fertigung angewandt wird, folgt dem DMAIC-Kreis, welcher im Wesentlichen an den Deming Kreis bzw. an den Plan-Do-Check-Act Circle basiert. DMAIC selbst besteht aus folgenden Phasen:

- *Definition (Definition)*
- *Messen (Measure)*
- *Analyse (Analyse)*
- *Verbesserung (Improvement)*
- *Kontrolle (Control)*

In der Produktentwicklung kommen abgewandelte Abläufe zum Einsatz, die zusammengefasst als Design for Six Sigma bezeichnet werden. Grundsätzlich folgt es aber dem gleichen Grundgedanken.

Interessant ist an dieser Stelle die Phase „Messen“, welche die tatsächliche Prozessleistungsfähigkeit ermittelt. Wie in allen Prozessverbesserungsaktivitäten gilt es den Prozessoutput zu verstehen und dessen Messung zu definieren. Was messen wir? Wie messen wir es am besten? Welche Daten sind verfügbar? Die in der „Define“ Phase erfassten Kundenbedürfnisse werden mit dem Kunden zusammen zu Messkriterien, Zielwerten und Spezifikationsgrenzen herunter gebrochen. Ergebnis ist ein sogenannter Critical to Quality (kurz CTQ) Baum. Eine Baumstruktur dient als Werkzeug zur Strukturierung ausgehend von den Kundenforderungen bis hin zu den messbaren Merkmalen. Ein weiteres Instrument ist Quality Function Deployment (QFD). Innerhalb eines Brainstormings werden den Merkmalen mehrere mögliche Messkriterien gegenübergestellt und jeweils priorisiert. Ziel ist eine minimale Menge an Kriterien, um den Messaufwand zu minimieren. In weiteren Schritten wird die Datenerfassung geplant. Wo liegen die Daten? Wie werden sie analysiert und dargestellt? Wie häufig werden sie erfasst? Wie werden sie erfasst? Für letzteres werden oft Formulare manuell ausgefüllt.

Im weiteren Verlauf werden die erfassten Daten für definierte Qualitätsmerkmale mittels Methoden der statistischen Prozesskontrolle ausgewertet. In der Regel kommt es zur Streuung bei Qualitätsmerkmalen. Diese Abweichungen werden in Beziehung zu den definierten Toleranzgrenzen des Qualitätsmerkmals gesetzt. Die als „Sigma“ bezeichnete Standardabweichung misst die Abweichung der Merkmalswerte. Je größer diese Standardabweichung, desto größer die Wahrscheinlichkeit einer Überschreitung der Toleranzgrenzen. Namensgebend für die „Six Sigma“ Methode ist nun, dass mindestens sechs Standardabweichungen zwischen dem Mittelwert aller Messungen und den Tole-

ranzgrenzen liegen sollen, um eine „Nullfehlerproduktion“ zu erzielen. Je kleiner der Abstand, desto höher die Wahrscheinlichkeit von Fehlern, wobei diese Korrelation auf Daten vergangener Läufe basiert.

Die statistischen Verfahren eignen sich vor allem für reproduzierbare Prozesse, wie dem Fertigungsprozess. Entwicklungsprojekte sind jedoch meist einzigartig in ihrem Ablauf. Qualitätsmerkmale sind über mehrere Projekte hinweg nicht identisch. Gerade die Frage, welche Qualitätsmerkmale relevant sind und wie eine effiziente Kalibrierung auf einzelne Projekte erfolgt, beantwortet Six Sigma nicht. Es sieht zwar eine Verfeinerung von Kundenanforderungen hin zu messbaren Qualitätsparametern vor. Eine informationstechnologische Umsetzung einer eindeutigen und einheitlichen Qualitätsdefinition fehlt jedoch. **Ziel 1 – Einheitliche Qualitätsdefinition** ist teilweise erfüllt. Die projektspezifische Qualitätsauswertung ist zwar gegeben, die Kalibrierung ist jedoch jedes Mal von neuem durchzuführen. Dies ist gerade in der Produktentwicklung mit ihren einzigartigen Projekten deutlich aufwendiger als in der Fertigung und dementsprechend wenig effizient. **Ziel 2 – Effiziente projektspezifische Qualitätsbewertung** ist nur teilweise erfüllt.

2.1.3 Quantitatives Messen

Im Bereich des Systems und Software Engineering sind Standards und Vorgehensmodelle entstanden, die Unternehmen bei der Etablierung quantitativer Messverfahren z.B. für den Bereich des Projekt- oder Qualitätscontrolling anleiten.

So beschäftigt sich z.B. das International Council of Systems Engineering kurz INCOSE mit diesem Thema. Laut INCOSE Systems Engineering Measurement Primer (INCOSE, 1998) ist Measurement ein Prozess, um einen Überblick über Fortschritt, Produkte und Prozesse eines Projekts bzw. des zu entwickelnden Systems zu erlangen. Dieser Überblick erhöht die Informationsbasis für Entscheidungsträger, deckt Abweichungen vom Plan auf, um frühzeitige korrektive Maßnahmen durchzuführen. Metriken sollten dabei keine fest definierte Menge darstellen. Vielmehr müssen Metriken auf die speziellen Probleme eines Projekts zugeschnitten sein, von denen es selten zweimal dasselbe gibt. Measurement umfasst daher Schritte zur Auswahl und Spezifikation von Metriken, der Erstellung eines Measurement Plans, Initialisierung und Ausführung der Datensammlung, Reporting der Ergebnisse und der Durchführung von Maßnahmen auf Basis der Ergebnisse. Diese Aktivitäten finden sich nahezu in allen Quellen, die Measurement Prozesse definieren und werden im Folgenden genauer beschrieben.

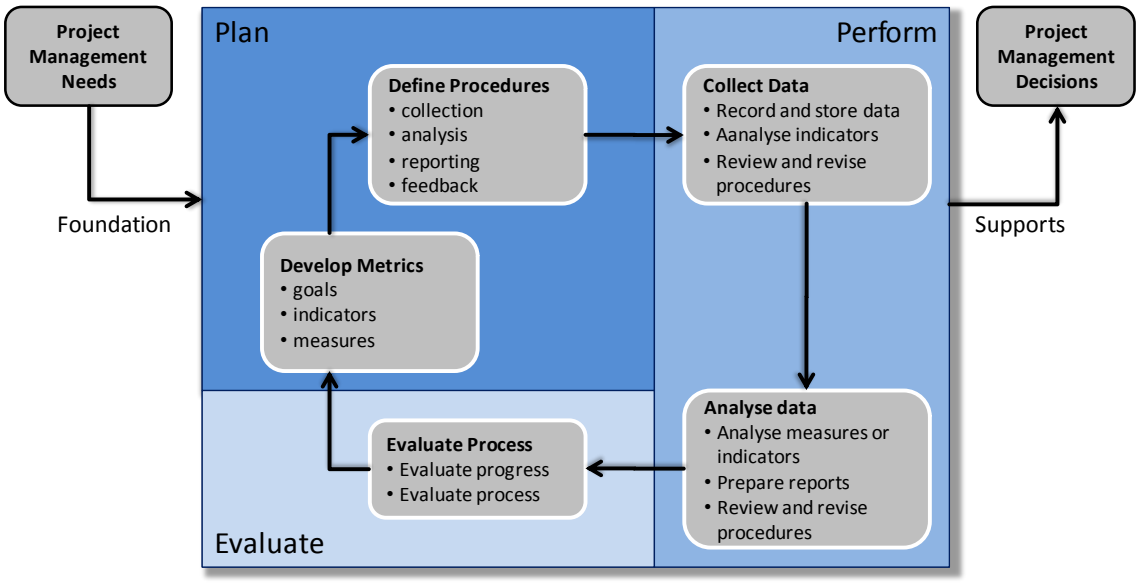

Abbildung 4: Übersicht Measurement Prozess (Park., 1996).

Die zwei in Abbildung 4 dargestellten Aktivitäten "Planung" und "Ausführung" stehen im Zentrum eines jeden Measurement Prozesses (siehe (Park, 1996), (ISO15939, 2007)). Zusätzlich enthalten Measurement Prozesse in der Regel eine Evaluierungsaktivität, um den Measurement Prozess und die verwendeten Metriken regelmäßig auf ihren Nutzen zu überprüfen und falls notwendig zu verbessern. Dies ist je nach Kapazitäten entweder qualitativ mit den Mitarbeitern oder aber unter Zuhilfenahme fortgeschrittener statistischer Methoden durchzuführen.

1. Planung des Measurement Prozesses

Die Planung definiert Gegenstand und Methode der Messung, sowie die Analysemethoden der erfassten Daten und die Aufbereitung durch das Reporting (INCOSE, 1998). Eine verbreitete Methode hierfür ist z.B. das Goal-Question-Metric (GQM) Paradigma von Victor Basili (Basili, 1988), (Basili, 1994), (Basili, 2002) angewandt.

1.1. Zieldefinition und Ableitung von Fragen

Die Planung beginnt mit wohldefinierten Zielen. Ziele sollten in einer verifizierbaren Form formuliert sein, von denen jedes Ziel in Fragen resultiert, wie die Zielerreichung messbar ist.

1.2. Definiere Metriken

Sind Fragen abgeleitet, gilt es Metriken zu definieren, welche die Informationen zur Beantwortung dieser Fragen liefern. Dabei ist der Anzahl von Metriken pro Frage keine Grenze gesetzt. Jeder Metrik werden Measurements zugeordnet, welche konkret messbare Daten darstellen. In einigen Fällen kann so ein Measure direkt die Metrik darstellen. In den meisten Fällen werden Metriken jedoch durch eine mathematische Kombination an Measures gebildet. Diese Datenaggregationen sind entsprechend zu definieren. Überall wo möglich sind diese Berechnungen zu automatisieren.

1.3. Definiere Datensammlung und Analyse

Um die Metriken messbar zu machen, ist ein in klar definierter und wiederholbarer Prozess zur Datensammlung und –analyse zu entwickeln. Dies muss mindestens folgendes umfassen:

- Welche Daten sind zu sammeln?
- Was sind die Quellen der zu sammelnden Daten?
- Wie werden die Messungen durchgeführt?
- Wer führt die Messungen durch?
- Wie oft werden Daten gesammelt?
- An wen werden die berechneten Metriken verteilt und in welchen Format?

2. Ausführung des Measurement Prozesses

Die folgenden Schritte werden während der Ausführung des Measurement Prozesses entsprechend Abbildung 4 durchgeführt:

2.1. Sammle und validiere Messdaten

Daten werden in regelmäßigen Zeitintervallen entsprechend der Datensammlungsprozeduren gesammelt. Die Daten werden validiert, indem die Korrektheit der Daten sichergestellt wird und keine Datenqualitätsprobleme vorliegen.

2.2. Berechne Metriken

Nach Validierung der Daten, werden Metriken und Indikatoren entsprechend der definierten Analyseprozeduren berechnet.

2.3. Verwalte Messungen und Metriken

Datenmessungen und Metriken sind in einer geeigneten Datenbasis zu speichern, um Konsistenz sicherzustellen und einen schnellen Zugriff zu gewährleisten.

2.4. Analysiere Messdaten und Metriken

Varianzen und Trends in den gesammelten Daten sind der Schlüssel zur Identifikation von Planabweichungen definierter Metriken. Trendanalysen helfen bei der Identifikation von Planabweichungen.

2.5. Verteilung von Analyseergebnissen

Sobald alle Analyseergebnisse verfügbar sind, gilt es diese Ergebnisse an alle relevanten Stakeholder zu verteilen. Sie sind die Grundlage für weitere Entscheidungsprozesse.

3. Evaluiere Measurement Prozess

Measurement Prozesse sind in ihren ersten Anwendungen selten optimal. Daher sind diese nach ihrer ersten Implementierung und in regelmäßigen Abständen zu evaluieren. Es ist zu prüfen, ob die Metriken ihren Zweck erfüllen und ob die Umsetzung ohne Probleme funktioniert. Ungenaue oder überflüssige Metriken sind entweder zu entfernen oder entsprechend anzupassen.

Folgende Bereiche sind im Idealfall Gegenstand der Evaluierung:

- Genauigkeit aktueller Metriken
- Überflüssigkeit von Metriken
- Behinderung der Projektarbeit durch definierte Messungen
- Genauigkeit von Analyseergebnissen
- Intervalle der Datensammlung und Analyse
- Vereinfachung von Metriken
- Projekt- oder organisationelle Änderungen

Zusammenfassend sind die beschriebenen Aktivitäten ein gutes Rahmenwerk für das quantitative Qualitätsmessen, jedoch eben nicht speziell auf diesen Bereich zugeschnitten. Solch generelle Rahmenwerke sind keine adäquate werkzeuggestützte Lösung, die gerade den Anforderungen der Kalibrierung auf einzelne Projekte gerecht wird. Die Notwendigkeit dieser Kalibrierung betonen die untersuchten Ansätze zwar, deren Durchführung lassen sie jedoch offen. Auf Grund des ähnlichen Vorgehens zum DMAIC Prozess in Six Sigma lassen sich ähnliche Zielerfüllungen ableiten. Obwohl allgemeiner als DMAIC und nicht direkt auf Qualität zugeschnitten, stellen die INCOSE Arbeiten und der Standard ISO/IEC 15939 auch eine Anleitung für eine einheitliche Qualitätsdefinition. Sie machen jedoch auch keine Hinweise, wie dies informationstechnologisch lösbar ist. **Ziel 1 – Einheitliche Qualitätsdefinition** ist daher teilweise erfüllt. **Ziel 2 – Effiziente projektspezifische Qualitätsbewertung** ist teilweise erfüllt, da eine projektspezifische Auswertung grundsätzlich vorgesehen ist. Hinter der Effizienz gerade im Kontext der Elektronikentwicklung steht jedoch auch hier ein großes Fragezeichen.

2.1.4 Diskussion

Ob Qualitätsmanagementansätze wie der PDCA Zyklus und Six Sigma oder die INCOSE Arbeiten und der Standard ISO/IEC 15939, alle beschreiben im Wesentlichen ein generelles Vorgehen zur Etablierung quantitativen Messens in einem Unternehmen. Obwohl letztere nicht ausschließlich für die Qualitätsmessung gedacht, lassen sie sich für das Qualitätsmonitoring anwenden. Diese typischen Vorgehen sind demzufolge auch für das Vorgehensmodell dieser Arbeit relevant. Sie sind jedoch allesamt zu generisch. Die Ansätze unterstützen zwar eine projektspezifische Bewertung, die initiale Kalibrierung ist aber sehr aufwendig. Es existieren keine Richtlinien, um eine gute Balance zwischen Wiederverwendbarkeit und Spezialisierung einer Qualitätsauswertung zu erreichen. Weiterhin enthalten die Ansätze keine Konzepte zur Anbindung an bestehende Entwicklungsprozesse, um die Auswertung auf Basis konkreter Entwicklungsdaten, wie z.B. Verifikations- und Validierungsaktivitäten durchzuführen. Software Project Control Center gehen hier weiter. So stellt das G-SPCC ein Framework dar, um Verfahren zur Qualitätsbewertung (z.B. Source Code Analyse) einzubinden. Jedoch fokussiert das Werkzeug Software und ist eher allgemein Projektcontrolling orientiert.

Die folgende Tabelle fasst die Zielerfüllung der Ansätze zusammen. X bedeutet „erfüllt“, (X) bedeutet „teilweise erfüllt“, keine Markierung bedeutet „nicht erfüllt“.

	Z1: Einheitliche Qualitätsdefinition	Z2: Effiziente Projekt-spezifische Auswertung	Z3: Integration V&V Ergebnisse	Z4: Domänenübergreifende Qualitätsbetrachtung	Z5: Entwicklungsbegleitende Auswertung	Z6: Berücksichtigung von Prozess- und Projektkontext	Z7: Automatische Datenerfassung und -integration	Z8: Anpassbarkeit an bestehende Entwicklungsprozesse
Klassische Projektmanagement Methoden						(X)		
Software Project Control Center					X		(X)	X
PDCA-Zyklus								
Six Sigma	(X)	(X)						
Measurement Standards	(X)	(X)						

Tabelle 1: Zielerfüllung Ansätze Projekt- und Qualitätscontrolling

Auf der Suche nach existierenden Arbeiten verlässt die Arbeit nun die eher Management orientierten Ansätze, um tiefer in die eigentliche Produktentwicklung und dort durchgeführten Aktivitäten einzutauchen. Die nächsten Abschnitte untersuchen daher den Stand der Technik aus den symbiotisch verwobenen Bereichen des Requirements Engineering und der Qualitätssicherung. Beide Bereiche bilden die Grundlage für eine entwicklungsbegleitende Qualitätsbewertung und einer Bestimmung des Ist-Werts für das Projektcontrolling.

2.2 Requirements Engineering

Anforderungen sind zentraler Maßstab bei der Bewertung, ob ein System eine ausreichende Qualität besitzt. Daher widmet sich dieses Kapitel dem Stand der Technik des Requirements Engineerings, bzw. bestimmten Themengebieten dieser Disziplin. Requirements Engineering ist wie folgt definiert (ISOWD29148.3, 2009):

Definition (Requirements Engineering) *Requirements Engineering ist eine interdisziplinäre Funktion zur Vermittlung zwischen Auftraggeber und Auftragnehmer, um die Anforderungen an das System, die Software oder die Dienstleistung aufzustellen und zu*

verwalten. Requirements Engineering beinhaltet die Identifikation, Erhebung, Spezifikation, Analyse, Bestimmung von Verifikationsmethoden, Validierung, Kommunikation, Dokumentation und die Verwaltung von Anforderungen.

Das Requirements Engineering ist damit ein zentraler Bestandteil eines jeden Entwicklungsprozesses und die Grundlage für eine Vielzahl an Maßnahmen zur Qualitätssicherung. Im Rahmen dieser Arbeit sind speziell die Anforderungsdefinition, sowie deren Verfolgung zu Implementierung und Verifikation relevant. Letzteres liefert eine Aussage über die Anforderungserfüllung und damit über die Qualität des Systems. Abschnitt 2.2.1 untersucht zu diesem Zweck Ansätze zur Modellierung funktionaler wie nichtfunktionaler Anforderungen. Abschnitt 2.2.2 beschreibt Ansätze für die Verfolgung von Anforderungen. Der letzte Abschnitt gibt schließlich eine Zusammenfassung und bewertet die Ansätze unter dem Gesichtspunkt der in dieser Arbeit angestrebten Ziele.

2.2.1 Anforderungsdefinition

Eine Qualitätsbewertung beantwortet die Frage „Erfüllt das System die definierten Anforderungen?“. Ist der Nachweis selbst zwar nicht Aufgabe der Anforderungsdefinition sondern der Verifikation, sind Anforderungen doch immer der Ausgangspunkt. Eine Evaluierung existierender Methoden zur Spezifikation von Anforderungen in Forschung und Praxis ist daher Fokus dieses Abschnitts. Dabei ist es das Ziel, die Diversität im Bereich der Anforderungsdefinition und die damit verbundene Schwierigkeit aufzuzeigen, die bei einer in dieser Arbeit angestrebten ganzheitlichen und transparenten Qualitätsbewertung besteht. Eine umfassende Übersicht und Beschreibung aller existierenden Anforderungsmodellierungsansätze würde den Umfang dieses Abschnitts sprengen und ist auch nicht Fokus dieser Arbeit. Dementsprechend liefert dieser Abschnitt eine kurze Übersicht verschiedener Methoden zur Anforderungsdefinition. Darauf aufbauend sind Ansätze zur Definition von Anforderungsmetamodellen im Fokus, die unter anderem das Ziel haben, die Diversität der existierenden Ansätze ein Stückweit Herr zu werden.

Existierende Anforderungsspezifikationssprachen unterscheiden sich in erster Linie in ihrem Formalisierungsgrad und damit in ihren Verifikations- und Validierungsmöglichkeiten. Darüber hinaus fokussieren einige Ansätze nur bestimmte Anforderungsarten, wie z.B. nicht funktionale Anforderungen. Die Bandbreite existierender Formalismen reicht von natürlich sprachlichen Anforderungen über Template basierte Ansätze wie z.B. Boilerplates (Hull, 2004) bis hin zu Methoden wie der UML Modellierung zur Definition von Aktivitätsdiagrammen. Schließlich existieren auch eine Reihe formaler Methoden, wie z.B. die Verwendung von Timed Automata (Alur, 1994) oder Software Cost Reduction (Heitmeyer, 2005), um nur einige zu nennen.

NFR Methode

Neben diesen vornehmlich zur Beschreibung funktionaler Anforderungen verwendeten Spezifikationssprachen existieren diverse Ansätze speziell zur Beschreibung nicht funktionaler Anforderungen, von denen eine Auswahl an dieser Stelle kurz umrissen wird. So beschreibt die NFR Methode von Dörr et al. (Dörr, 2005) mit Hilfe von Qualitäts-

modellen Abhängigkeiten und Verfeinerungsrelationen zwischen nicht funktionalen Qualitätscharakteristika wie z.B. Effizienz oder Wartbarkeit. Eine nicht funktionale Anforderung beschreibt einen Zielwert für eine Qualitätscharakteristik in einem konkreten Projekts. Die definierten Qualitätsmodelle dienen dabei als Guideline für die Anforderungsdefinition. Sie sind Grundlage für die Struktur von Checklisten und Templates. Abhängigkeiten zwischen Eigenschaft werden in Qualitätsmodellen gespeichert und müssen nicht für jedes Projekt neu definiert werden.

Abbildung 5 beschreibt einen Ausschnitt des Metamodells aus der NFR Methode (Dörr, 2003). Es definiert Relationen („refined into“, „influences“) zwischen Qualitätsattributen, zur Beschreibung von Qualitätsmodellen. Ferner beschreibt es Verknüpfungen zwischen nicht funktionalen Anforderungen und Qualitätscharakteristika (hier Qualitätsattribute genannt).

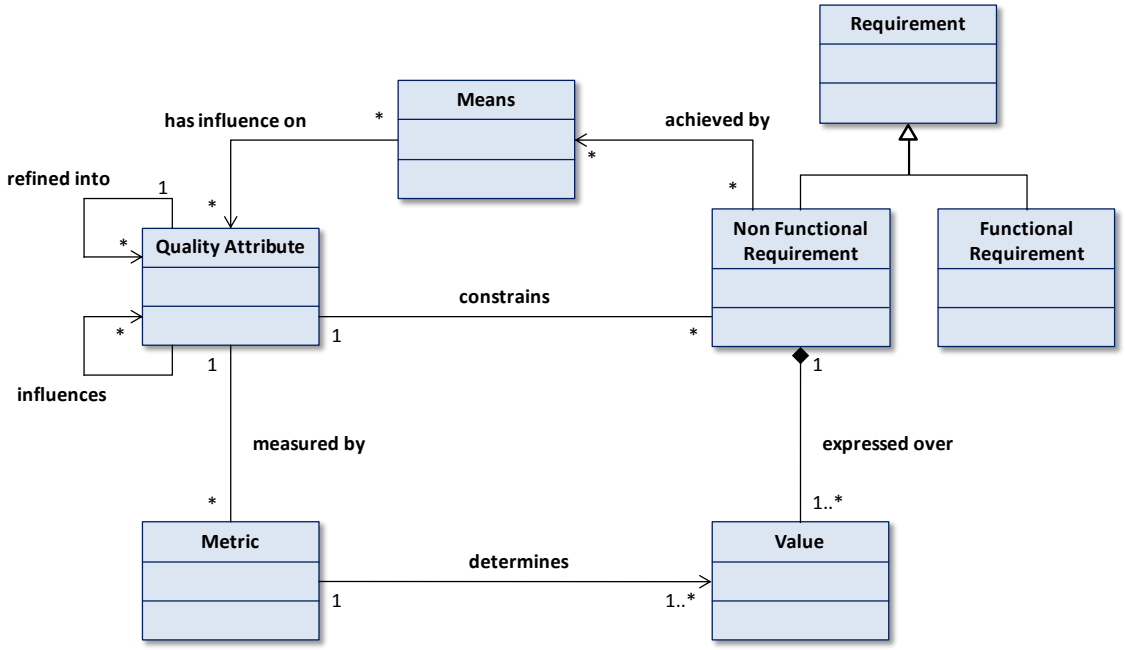

Abbildung 5: Ausschnitt des Metamodells aus (Dörr, 2003)

Definition nicht funktionaler Anforderungen basierend auf aktivitätsbasierten Qualitätsmodellen

Ähnlich zum vorherigen Ansatz erheben und verfeinern Wagner et al. (siehe (Wagner, 2007b), (Wagner 2008)) Anforderungen entsprechend der durch Qualitätsmodelle vorgegebenen Struktur. Grundsätzlich auch hier mit dem Ziel, quantitative Anforderungen als Ergebnis zu erhalten. Der Unterschied zum vorherigen Ansatz besteht lediglich in der Art der Qualitätsmodelle. Hier verwenden Wagner et al. aktivitätsbasierte Qualitätsmodelle (siehe Abschnitt 2.3.2 dieser Arbeit).

NoFun

Ein weiterer Ansatz namens NoFun (siehe (Franch, 1998), (Botella, 2001)) stellt eine Spezifikationsmöglichkeit nicht funktionaler Eigenschaften für Software Komponenten dar. Auch hier dienen Qualitätsmodelle als Werkzeug zur Verfeinerung abstrakter hin

zu konkret messbaren Eigenschaften. Diese Eigenschaften inkl. Zielwerte werden mit Hilfe des NoFun Formalismus spezifiziert. Auch wenn hier eine eigene Spezifikationsmethodik von Qualitätseigenschaften eingeführt wird, so ähnelt dieser Ansatz doch ebenfalls den vorangegangenen.

Trotz dieser Vielzahl an Spezifikationsmöglichkeiten sind in der Praxis natürlich sprachliche Anforderungsdefinitionen z.B. mit Microsoft Word, Excel oder IBM Rational DOORS[4] immer noch am Verbreitesten. Jedoch ist die Verwendung einer bestimmten Methode keinesfalls ein Ausschlusskriterium für andere Anforderungsspezifikationssprachen. Vielmehr ergänzen sich die verschiedenen Formalismen, so dass je nach Entwicklungsphase die Anwendung verschiedener Ansätze in Kombination sogar von Vorteil ist, um insgesamt ein besseres Verständnis des Problemraums zu erlangen. Trotzdem – oder gerade deshalb – bleibt es eine Herausforderung alle Arten von Anforderungen unter Berücksichtigung der verschiedenen Formalisierungsgrade zusammenzuführen, um alle notwendigen Qualitätscharakteristika zu berücksichtigen.

Um die verschiedenen existierenden Formalismen auf eine zumindest strukturell einheitliche Basis zu heben und natürlich sprachlichen Anforderungen mehr Struktur zu geben, setzen verschiedene Ansätze auf die Verwendung von Metamodellen. Im Folgenden werden verschiedene modellbasierte Ansätze zur Beschreibung von Anforderungen vorgestellt.

SysML

SysML (SysML, 2009) ist eine graphische Modellierungssprache für das Systems Engineering und erweitert die UML 2.0 (UML, 2005). Wie in Abbildung 6 dargestellt, dient SysML ist in erster Linie zur Beschreibung von Struktur und Verhalten von Systemen. Die Sprache bietet hierfür verschiedene Diagrammtypen an.

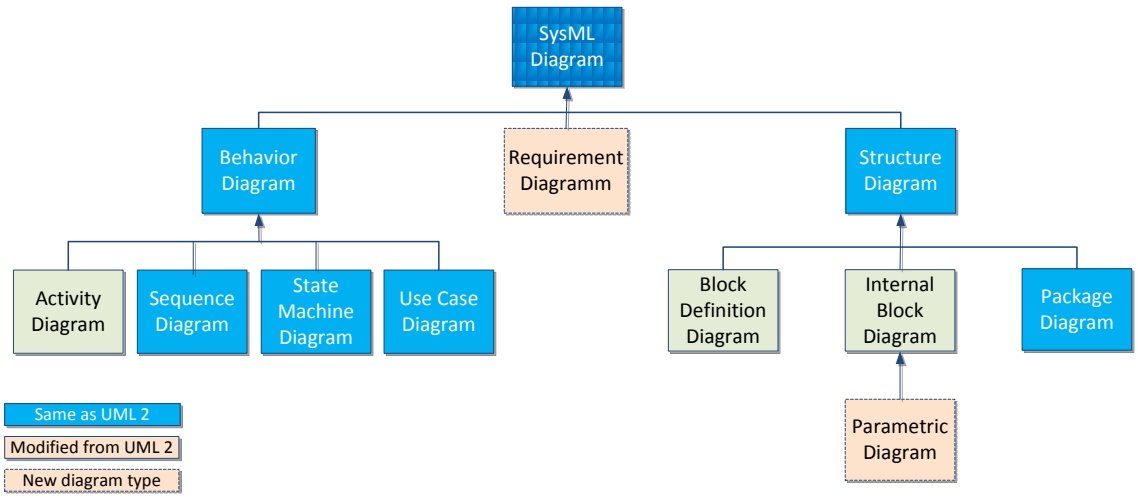

Abbildung 6: SysML Diagramme

[4] Siehe http://www-01.ibm.com/software/awdtools/doors/

Neben den genannten Diagrammen, von denen die meisten aus der UML übernommen bzw. angepasst wurden, wird durch SysML auch ein neues Diagramm zur Beschreibung von Anforderungen (Requirements Diagram) eingeführt (siehe Abbildung 7). Die Definition von Anforderungen selbst geht dabei nicht über die ID, einen beschreibenden Text und die Einordnung entsprechend selbst definierter Kategorien (z.B. Funktional oder Performance) hinaus. Fokus liegt daher weniger auf einer detaillierten Anforderungsdefinition, sondern vielmehr auf der Verknüpfung von Anforderungen mit anderen Design Artefakten.

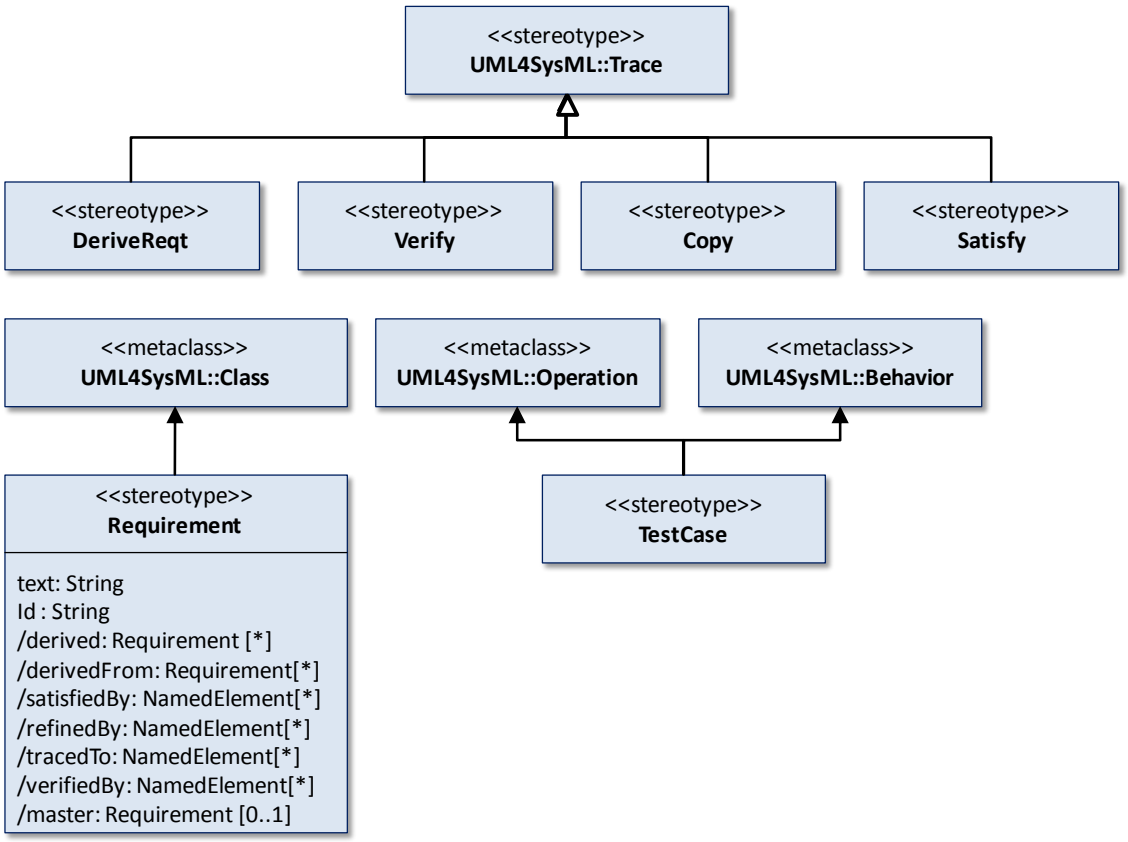

Abbildung 7: SysML Konzepte des Requirements Diagram

Anforderungsverfolgungskonzepte in SysML decken die Verknüpfung von Anforderungen mit Systemkomponenten („Satisfy“), Testfällen („Verify“) und Anforderungen untereinander ab („DeriveReqt“, „Refine“). Abschnitt 2.2.2 beschreibt im Detail, was sich hinter Konzepten der Anforderungsverfolgung verbirgt.

EAST-ADL2

Bei EAST-ADL2 handelt es sich um eine Architekturbeschreibungssprache für die Domäne Automotive. Sie stellt das Ergebnis der Projekte EAST-EEA, ATESST (ATESST, 2008) und ATESST 2 (ATESST, 2010). Das Metamodell erlaubt die Definition von Systemarchitekturen, sowie Software und Hardware Komponenten, Kommunikation zwischen Komponenten und die Definition der Umgebung in der sie eingesetzt werden.

Orthogonal zur Beschreibung von Architekturen besteht die Möglichkeit der Beschreibung von Anforderungen sowie von Verifikation und Validierung.

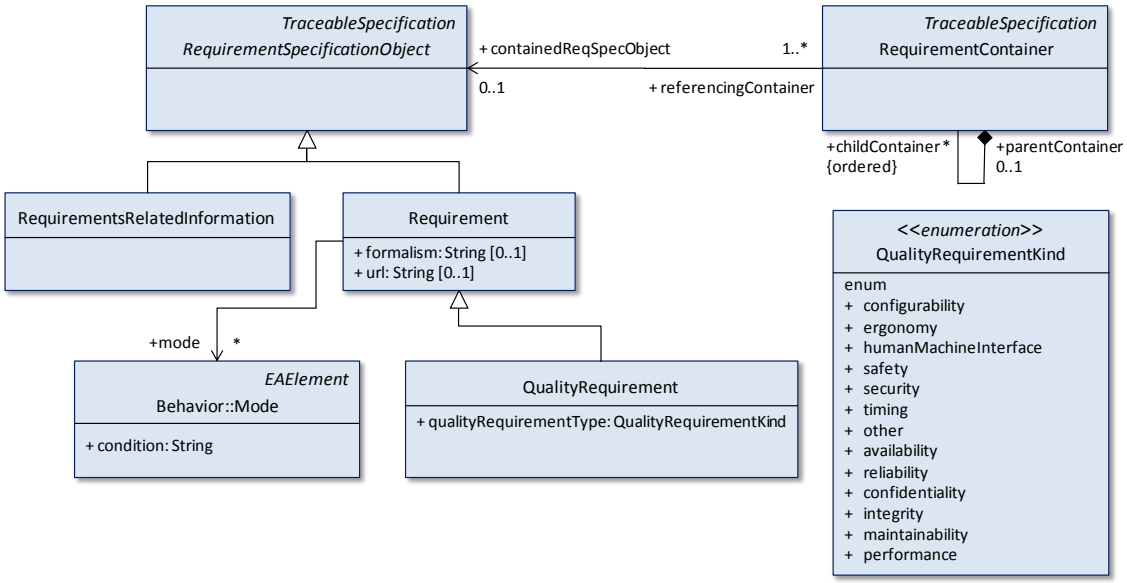

Abbildung 8:Anforderungskonzepte aus EAST-ADL2 (ATESST, 2010)

Wie Abbildung 8 zeigt, enthält EAST-ADL nicht signifikant mehr Attribute und Konzepte zur Beschreibung von Anforderungen als SysML. Es besitzt Konzepte zur Beschreibung von Qualitätsanforderungen und eine fest vorgegebene Anzahl an Kategorien.

Darwin4Req

Ein weiterer modellbasierter Ansatz zur Beschreibung von Anforderungen ist DARWIN4Req (Canilla, 2010), ein im Forschungsprojekt MeMVaTex (Albinet, 2008) entwickeltes UML Profil. Das Profil fokussiert sich auf die Beschreibung von Anforderungen und führt im Vergleich zu den vorher beschriebenen Ansätzen weitere Attribute für Anforderungen ein. Das Profil wird unter anderem auch im ReMIAS (Adejouma, 2010) Ansatz verwendet, welches das Darwin4Req Profil mit dem EAST-ADL2 Profil kombiniert und erweitert, um Anforderungen des Automotive Sicherheitsstandards ISO26262 (ISO26262, 2010) hinsichtlich der Beschreibung von Anforderungsqualitätsattributen zu ermöglichen. Bis auf die zusätzlichen Attribute, weisen jedoch sowohl das Darwin4Req als auch der darauf aufbauende ReMIAS Ansatz keine wesentlichen Unterschiede zu EAST-ADL2 und SysML auf.

Göknil et al.

In (Göknil, 2008) präsentieren die Autoren ebenfalls einen Metamodell basierten Ansatz. Auf Grund der verschiedenen existierenden Ansätze, die teilweise domänen- und unternehmensspezifisch sind, wird ein generisches Metamodell notwendig, um unabhängig von der konkreten Ausprägung im Unternehmen bestimmte Analysen durchzuführen. Im Falle der Autoren ist es das logische Schließen über Abhängigkeiten zwi-

schen Anforderungen. Dabei haben sie kein Interesse, alle Konzepte bis hin zu einer detaillierten Anforderungsspezifikation wie z.B. für Realzeitanforderungen eingebetteter Systeme abzubilden, sondern vielmehr gemeinsame Konzepte verschiedener bereits existierender Anforderungsmodellierungstechniken in einem Core Metamodell zusammenzufassen. Domänenspezifische Sprachen zur Beschreibung von Anforderungen gibt es viele, die jedoch dann an ein solches Core Metamodell anzubinden sind.

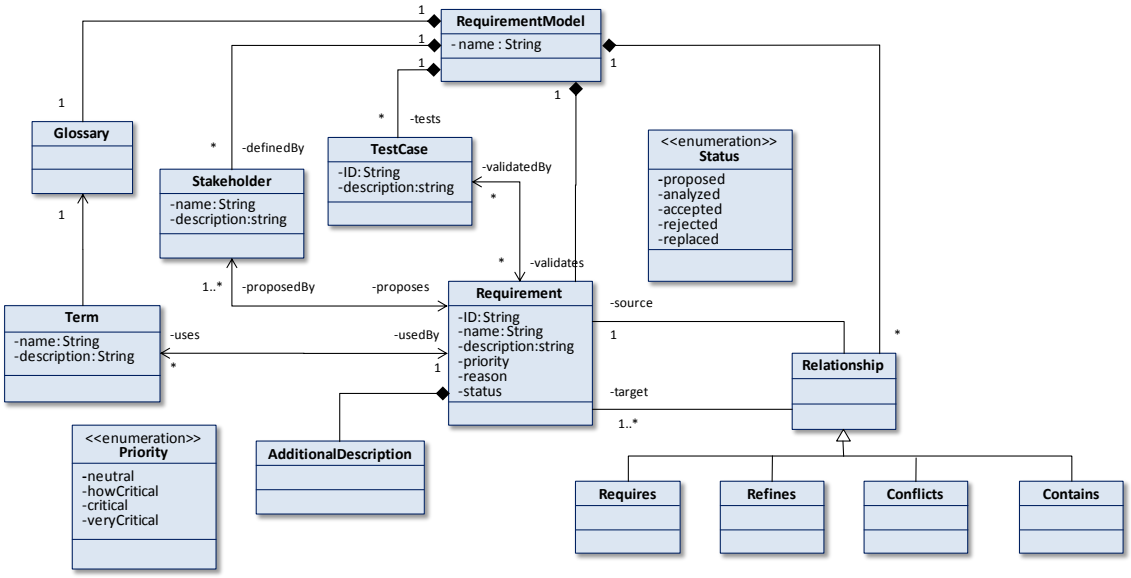

Abbildung 9: Metamodell nach Göknil (Göknil, 2008)

Das entwickelte Metamodell (dargestellt in Abbildung 9) enthält z.B. Konzepte zur Beschreibung von Anforderungsmodellen, Anforderungen und Testfällen. Weiterhin definiert es die typischen Attribute und mögliche Relationen zwischen Anforderungen (siehe „Relationship“ und Unterkonzepte). Auf eine, wie in anderen Ansätzen oftmals typische Anforderungskategorisierung verzichten sie, da es hierfür je nach Domäne jeweils unterschiedliche Vorstellungen gibt und daher Bestandteil von Erweiterungen sind. Am Beispiel von SysML demonstrieren sie jedoch wie das Core Metamodell mit einer Menge von Anforderungskategorien erweiterbar ist.

Selbst bei Existenz eines den Anforderungen entsprechenden Metamodells, reicht eine Definition von Anforderungen allein nicht aus, um eine projektspezifische Qualitätsdefinition zu erlangen. Notwendig ist darüber hinaus eine eindeutige Allokation der Anforderungen an das System, welches diese umsetzt. Der folgende Abschnitt widmet sich diesem Thema.

2.2.2 Anforderungsverfolgung

Anforderungsverfolgung (engl. requirement traceability) dient im Wesentlichen einem Zweck: das Nachhalten von Ergebnissen in der Verarbeitung und Verfeinerung von Anforderungen während des Requirements Engineering und deren Umsetzung entlang der gesamten Produktentwicklung. Anforderungsverfolgung ist Aufgabe des Require-

ments Managements, einer Teildisziplin des Requirements Engineerings. Gotel (Gotel, 1994) definiert Anforderungsverfolgung wie folgt:

Definition (Anforderungsverfolgung) *Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction, i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases.*

Anforderungsverfolgung bezieht sich also auf die Möglichkeit das gesamte "Leben" einer Anforderung zu verfolgen, von ihrer Erzeugung über ihre Umsetzung bis hin zu allen durchgeführten Änderungen. Der Artikel führt darüber hinaus die Begriffe "pre-requirements specification" (pre-RS) traceability und "post-requirements specification" (post-RS) traceability ein. Pre-RS traceability beschreibt die Verfolgung von Anforderungen während ihrer Erhebung, also ihrer Spezifikation und Verfeinerung. Darüber hinaus gehört hierzu die Dokumentation der Stakeholder von Anforderungen, ihrer Quellen, der Grund ihrer Existenz und das Verfolgen von Änderungen. Auf der anderen Seite beschreibt Post-RS Traceability die Verfolgung von Anforderungen zu den sie umsetzenden System Komponenten sowie Tests und Analysen, welche sie verifizieren.

Ziele Existierende Literatur (z.B. (Watkins 1994), (Brcina, 2008)) beschreibt verschiedene Ziele und Anwendungen, welche eine systematische Anforderungsverfolgung motivieren. Jedes dieser Ziele und Anwendungen haben ihre eigenen Anforderungen an die Ausprägung der Anforderungsverfolgung. So hat es z.B. einen Einfluss, wer die Informationen verwendet (Kunde, Projektmanager, Entwickler, Test Ingenieur) und welche Entwicklungsphase (Anforderungserhebung, Architekturentwicklung, Implementierung, Test, Betrieb, etc.) betrachtet wird. Im Folgenden sind verschiedene Anwendungsfelder genauer beschrieben:

- *Projektmanagement:* Das Management eines Projekts umfasst eine Vielzahl an Aufgaben von denen Antworten auf die Fragen "Wann sind wir fertig?" und "Wie teuer wird es werden?" einige der relevantesten sind. Anforderungsverfolgung hilft bei der Analyse des aktuellen Projektstatus „Was ist schon fertig, was ist noch zu tun?". Dieser Punkt hat daher eine starke Verknüpfung zur Bestimmung der aktuellen Produktqualität.
- *Änderungsmanagement:* Während des Verlaufs eines Projekts unterliegen Komponenten und Anforderungen immer wieder Erweiterungen und Änderungen basierend auf neuen Erkenntnissen während der Entwicklung oder neuen Kundenanforderungen. Unabhängig des Grunds ist die potentielle Auswirkung einer Änderung auf Komponenten und andere Anforderungen zu identifizieren, um weitere notwendige Anpassungen, Rezertifizierung oder erneut durchzuführende Tests zu erkennen. Diese Analyse wird auch als Einflussanalyse (engl. Impact Analysis) bezeichnet.

- *Validierung:* Eine typische Fragestellung in der Praxis bezieht sich auf die Frage der Vollständigkeit meiner Anforderungen: „Habe ich alle notwendigen Anforderungen erfasst?“. Eine mögliche Sichtweise wäre in diesem Zusammenhang die Frage „Habe ich Anforderungen vergessen, die für mein Produkt und dessen Fähigkeiten noch relevant wären?“. Diese Frage ist nur schwer zu beantworten und nach besten Wissen und Gewissen mit Hilfe des Kunden zu erarbeiten. Eine andere Herangehensweise an die erste Frage ist “Decken meine Anforderungen alle Anforderungen der höheren Ebene ab?”. Anforderungsverfolgung liefert eine Antwort auf diese Frage. Ebenso liefert es eine Antwort auf die Frage “Besitzen alle Anforderungen der aktuellen Ebene eine zugehörige auf der höheren Ebene?”, um unnötige Anforderungen und Funktionalitäten zu erkennen.
- *Verifikation:* Verifikation überprüft, ob alle Anforderungen durch die Implementierung erfüllt sind. Mittels Anforderungsverfolgung wird sichergestellt, dass alle Anforderungen und Systemkomponenten durch eine ausreichende Anzahl an Testfällen abgedeckt werden.
- *Design*: Anforderungsverfolgung dient der Dokumentation und Nachhaltung von Alternativen und Entscheidungen, die während der Entwicklung getroffen wurden.
- *Zertifizierung:* Anforderungsverfolgung ist Grundlage für eine klare und eindeutige Systemdokumentation, die unabdingbar für die Zertifizierung und den Nachweis der Standardkonformität sind. Dies spielt insbesondere bei der Entwicklung sicherheitskritischer Systeme eine wichtige Rolle.

Definitionen Referenz bzw. Metamodelle für die Anforderungsverfolgung sind in aller Regel die Basis existierender Methoden zur Erzeugung und Management von Anforderungsverfolgungsinformationen. Im Allgemeinen definieren diese Modelle:

(a) Welche Design Artefakte während der Entwicklung verfolgt werden sollen.

(b) Welche Arten an Relationen zwischen den verschiedenen Design Artefakt Typen erzeugbar sind.

Dieser Absatz stellt eine Auswahl bekannter Metamodelle vor, die in Zusammenhang mit dem Thema Anforderungsverfolgung entstanden sind:

Referenzmodell nach Ramesh und Jarke

Ein prominentes und viel zitiertes Referenzmodell für Anforderungsverfolgung stellt das von Ramesh und Jarke (Ramesh, 2001) dar. Basierend auf einer empirischen Studie präsentieren die beiden Autoren ein Referenzmodell, welches Best Practices von 26 amerikanischen Software Firmen bzgl. implementierter Anforderungsverfolgungslösung darstellt.

Das Referenzmodell umfasst zwei Arten von Modellen: eines für „low-end traceability user“ und eines für „high-end traceability user”. Eine Unterscheidung in diese beiden Benutzergruppen basiert im Wesentlichen auf deren Erfahrung (null bis zwei Jahre ge-

genüber 5 bis 10 Jahre) und der Komplexität der abgewickelten Projekte (1000 Anforderungen pro Projekt gegenüber 10.000 Anforderungen).

"Low-End" Anwendung Anforderungsverfolgung

Typische "low end" Benutzer (siehe Abbildung 10) betrachten Anforderungsverfolgung als eine Verknüpfung von Anforderungen zu den sie erfüllenden System Komponenten. Basierend darauf ist verifizierbar, dass jede Anforderungen an eine Komponente allokiert ("allocated_to") und jede Funktion/Komponente wiederum existierende Anforderungen umsetzt ("satisfy").

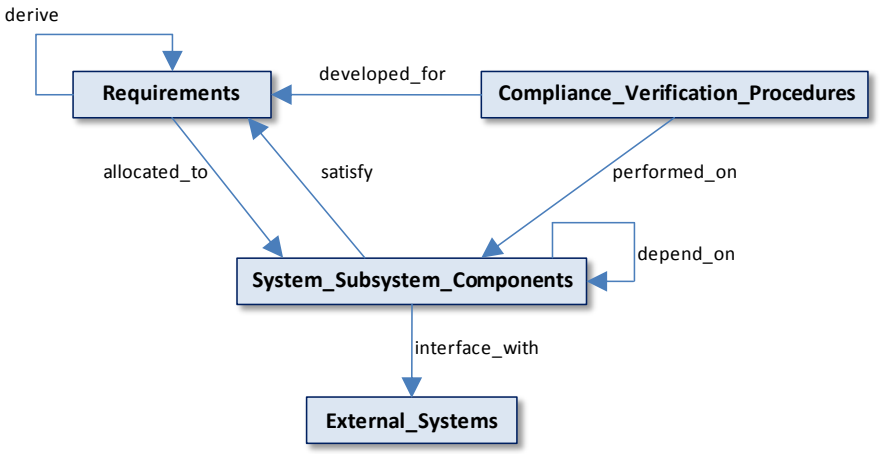

Abbildung 10: Low-End Traceability Model

Abstraktere Systemanforderungen werden zu detaillierteren Anforderungen verfeinert. Verknüpft werden solche Anforderungen mit Hilfe der „Derive" Relation. Ob eine Komponente auch wirklich die mit ihr verknüpfte Anforderung erfüllt ist Aufgabe des Testens. Für jede Anforderung werden Verifikationsprozeduren (Tests, Simulationen) entwickelt und mit der entsprechenden Anforderung und Komponente verknüpft. Verifikationsergebnisse geben Hinweise über die Erfüllung der Anforderung. Weiterhin sind Abhängigkeiten zwischen Komponenten definierbar.

„High-End" Anwendung Anforderungsverfolgung

Das „High-End" Modell teilt sich in drei Sub-Modelle auf: Requirements Management, Design Allocation und Compliance Verification. Des Weiteren ist Rationale Management ein zentraler Aspekt im Vergleich zum „Low-End" Modell, dem dieses fehlt.

Das Requirements Management Modell in Abbildung 11 erlaubt die Definition von Organisationszielen („Organizational Needs"), die durch Szenarien eine Detaillierung erfahren. Aus diesen Szenarien lassen sich allgemeine Systemziele („System Objectives") ableiten. Darüber hinaus erlaubt das Modell die Verfolgung von Anforderungsänderungen und Änderungsanfragen ("Change Proposals"). Ferner führt das Modell Verknüpfungen wie 'elaborate' (Klarstellung von Anforderungen) oder 'depend on' ein (die Anforderung existiert nur auf Grund einer anderen bereits existierenden). Da es in großen Projekten nicht ratsam ist, alles nachzuverfolgen, helfen „Critical success factors"

(CSR) bei der Gewichtung von Anforderung und der Priorisierung des Anforderungsverfolgungsaufwands.

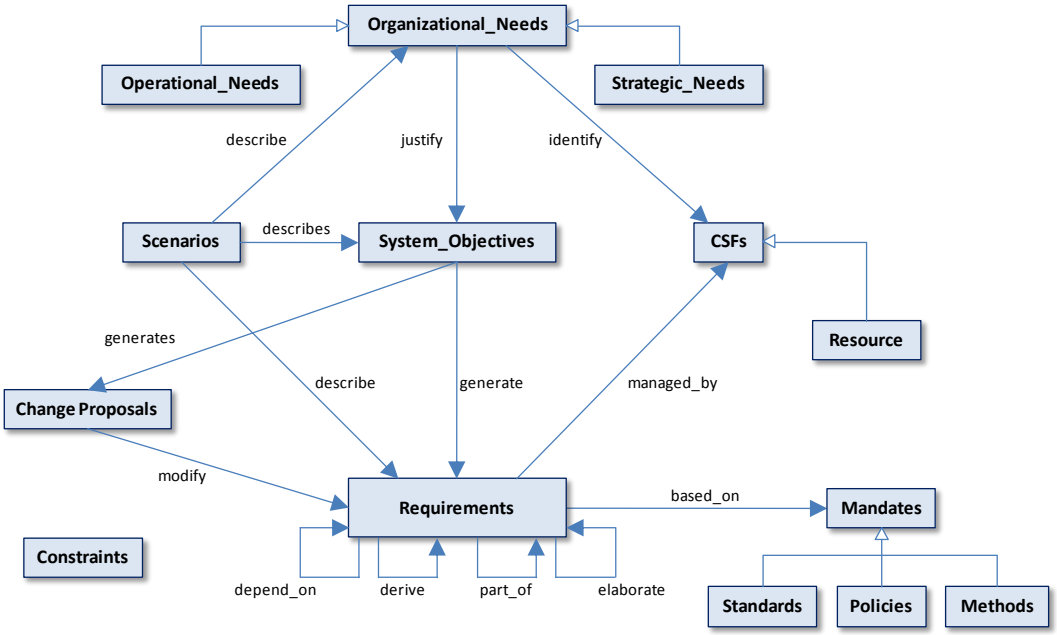

Abbildung 11: High-End Traceability Model – Requirements Management Model

Ziel des in Abbildung 12 dargestellten Rationale Modells ist die Dokumentation von Gründen der Existenz oder Änderung einer Anforderung, sowie von Design Entscheidungen. Dies erlaubt darüber hinaus auch die Dokumentation nicht weiter verfolgter Alternativen und abgelehnten Anforderungen.

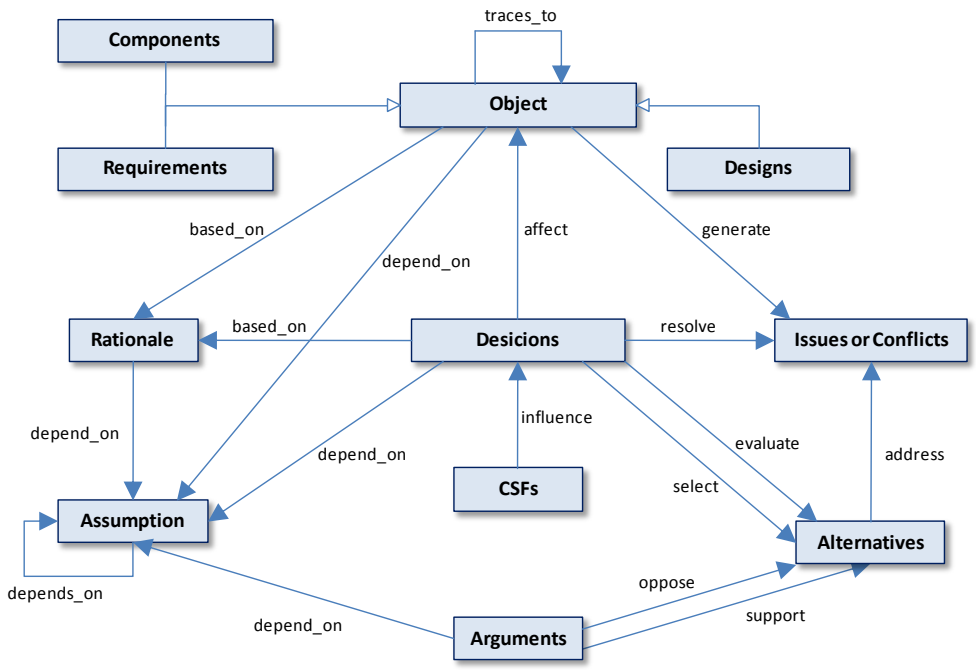

Abbildung 12: High-End Traceability Model – Rationale sub-model

Das “Design allocation” Modell definiert Verknüpfungen zwischen Anforderungen und Komponenten und führt im Vergleich zum „Low End“ Modell weitere Verknüpfungen zwischen Komponenten, wie die part-of Beziehung ein. Das letzte Modell in derselben Abbildung ist das „Compliance Verification“ Modell. Es werden spezielle Konzepte wie Tests, Prototypen, Inspektionen oder Simulationen zusammen mit Verknüpfungen zu Standards und Richtlinien eingeführt, welche der Grund für die Durchführung der Verifikation ist. Beide Modelle stellt Abbildung 13 dar.

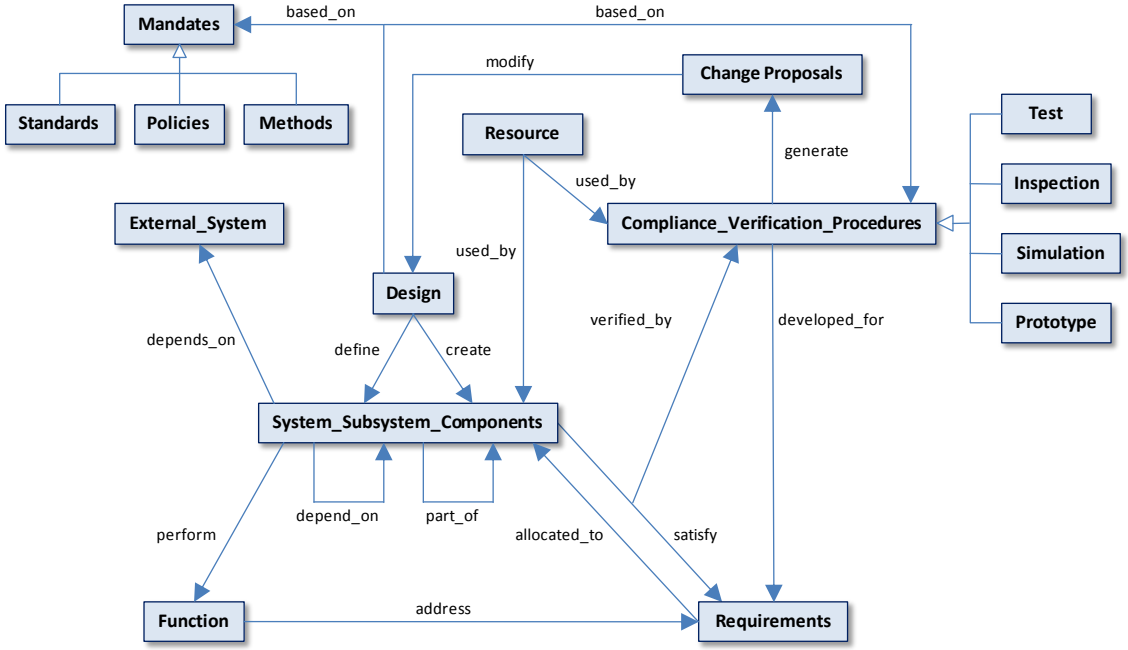

Abbildung 13: High-End Traceability Model – Design allocation and compliance verification sub-model

Für eine detaillierte Beschreibung dieser vier Modelle sei auf (Ramesh, 2001) verwiesen. Auch wenn man sich im Detail über Bedeutungen streiten mag, so zeigt das Modell von Jarke and Ramesh, auf welche Gebiete bei der Anforderungsverfolgung zu achten ist.

TraceChange

Von Knethen (von Knethen, 2001a), (von Knethen, 2001b), (von Knethen, 2002) entwickelte einen so genannten änderungsorientierten Anforderungsverfolgungsansatz für eingebettete Systeme mit Fokus auf Änderungen funktionaler Anforderungen. Ziel ist die Etablierung einer Grundlage für aussagekräftige Einflussanalysen von Änderungen. Die entwickelte Architektur besteht aus drei Ebenen:

Conceptual System Model:

Das konzeptionelle System Model beschreibt die Entitäten und deren Beziehungen untereinander, die während der Anforderungsdefinition und funktionalen Beschreibung des Systems erzeugt werden. Die Beschreibungen sind dabei unabhängig von der konkret verwendeten Sprache bzw. Dokumentformat.

Conceptual Documentation Model:

Das konzeptionelle Dokumentationsmodell beschreibt Notationen und Entitäten, die zur Definition von Anforderungen verwendet werden. Nach (von Knethen, 2001b) beschränkt sich TraceChange jedoch auf Entwicklungsdokumente zur Beschreibung von Systemanforderungen, Softwareanforderungen und der Softwarearchitektur.

System Requirements:

Die letzte Ebene beschreibt schließlich die Anforderungen selbst, die in den verschiedenen Dokumenten enthalten sind. Es besteht die Möglichkeit, Elemente zwischen dem Dokumentations- und dem System Modell zu verknüpfen. Basierend auf Abhängigkeiten im System Modell sind Abhängigkeiten im Dokumentationsmodell automatisch erkennbar. Existieren Verfeinerungsrelationen zwischen zwei Funktionen, werden auch die Paragraphen im korrespondierenden Dokumentationsmodell miteinander verknüpft. Umgesetzt wurde der Ansatz mittels eines DOORS Add-On.

EAST-ADL2

EAST-ADL2 enthält neben den bereits in Abschnitt 2.2.1 beschriebenen Konzepten zur Abbildung von Anforderungen ebenfalls Konzepte zur Anforderungsverfolgung. Abbildung 14 zeigt das Diagramm aus (ATTEST, 2010) mit Anforderungsverfolgungskonzepten aus EAST- ADL2.

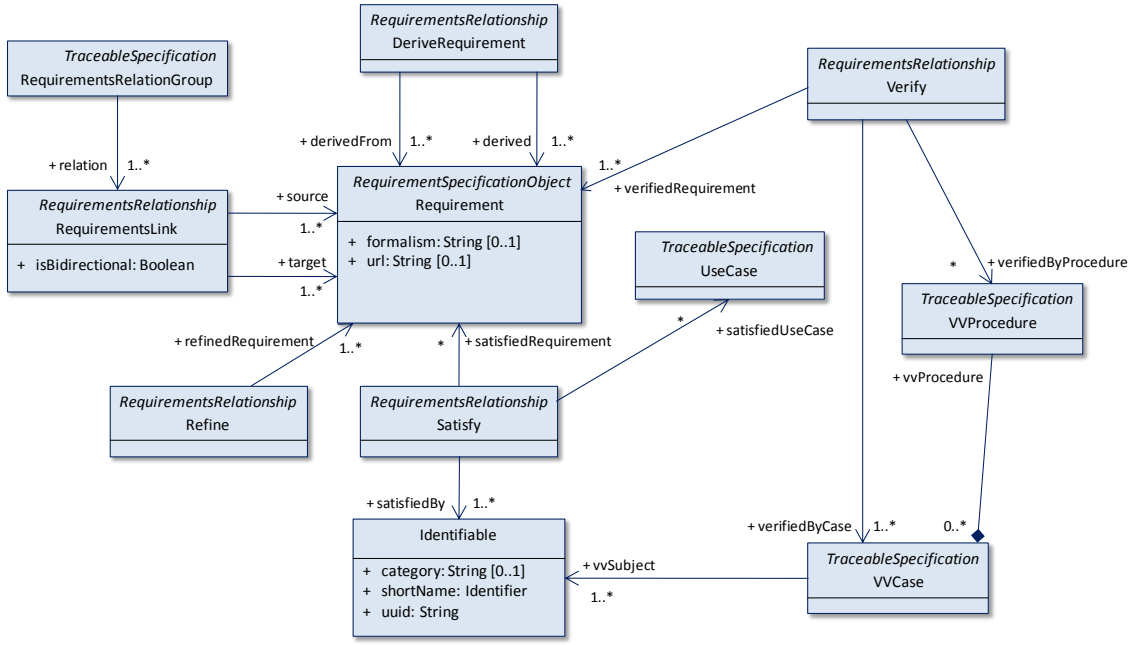

Abbildung 14: Anforderungsverfolgungsrelationen in EAST-ADL2

Anforderungsverfolgung in EAST-ADL2 folgt den Prinzipien von SysML: EAST-ADL2 ermöglicht die Verfolgung der Dekomposition und Verfeinerung von Anforderungen innerhalb einer Abstraktionsebene und zwischen mehreren Abstraktionsebenen. Zu diesem Zweck existieren die Verknüpfungen „DerivedRequirement“ und „Refine“. Darüber hinaus ermöglicht EAST-ADL2 die Beschreibung von Tests und Analysen zur

Verifikation und Validierung von Anforderungen sowie der Verknüpfung von Tests mit Anforderungen und Systemkomponenten, die sie analysieren und verifizieren. Zu diesem Zweck existiert die „Verify“ Verknüpfung. Zuletzt werden Anforderungen an Systemkomponenten mit Hilfe der „Satisfy“ Verknüpfung allokiert.

Methoden und Werkzeuge zur Erzeugung von Verknüpfungen Ist die Menge an gültigen Verknüpfungen mittels eines Metamodells definiert, gilt es die Erzeugung dieser Verknüpfungen zu unterstützen. Zu diesem Zweck sind in der Vergangenheit Methoden und Werkzeuge entstanden, die im Folgenden kurz vorgestellt werden. Existierende Methoden lassen sich in die folgenden Bereiche einteilen. Auf eine detaillierte Beschreibung der einzelnen Bereiche sei auf die jeweils genannten Referenzen verwiesen:

- *Information Retrieval:* Erzeugt Verknüpfungen zwischen zwei Objekten basierend auf der Ähnlichkeit ihrer Bezeichner. Es existieren eine Reihe verschiedenster Ansätze in diesem Bereich, die sich hauptsächlich in ihrer Genauigkeit unterscheiden. Für eine Übersicht sei auf de Lucia (de Lucia, 2009) verwiesen.
- *Wert-basierte Anforderungsverfolgung:* Unterscheidet zwischen Anforderungen, für die sich die Verfolgung lohnt und für welche nicht, um den Aufwand für die Anforderungsverfolgung gering zu halten. Kriterien sind z.B. Relevanz für Stakeholder, Risiko oder Volatilität (siehe z.B. (Heindl, 2005), (Egyed, 2005a), (Egyed, 2005b)).
- *Regel basierte Ansätze:* Spanoudakis et al. (Spanoudakis, 2002), (Spanoudakis, 2004), und Zisman et al. (Zisman, 2002), (Zisman, 2003) stellen Ansätze zur Erzeugung von Verknüpfung basierend auf definierten Regeln vor.
- *Szenario basierte Ansätze:* In (Egyed, 2003) präsentiert Egyed einen Szenario basierten Ansatz zur Verknüpfung von Software Komponenten mit UML Diagrammen und benötigt ein ausführbares System, dessen Verhalten in definierten Test Szenarien überwacht wird. Basierend auf so gesammelten Laufzeit Informationen (ausgeführte Klassen) werden Verknüpfung abgeleitet und validiert.
- *Ereignis basierte Ansätze:* Cleland-Huang et al. (Cleland-Huang, 2002), (Cleland-Huang, 2003) erzeugen Verknüpfungen zwischen quantitativen Performance Anforderungen und Variablen in ausführbaren Modellen, welche Anforderungen bzgl. Antwortzeiten und Durchsatz validieren. Stärke dieses Ansatzes ist die direkte Evaluierung von Anforderungsänderungen auf die Performance des Systems.

Neben diesen vorwiegend akademischen Ansätzen existieren verschiedene kommerzielle Werkzeuge mit denen man (meist manuell) Verknüpfungen erzeugen kann. So erlaubt es z.B. das Anforderungsmanagement Werkzeug DOORS mögliche Verknüpfungen zu definieren und diese auch zu erzeugen. In der Elektronik- und Softwareentwicklung sind auf DOORS basierende Implementierung der Anforderungsverfolgung weit verbreitet. Diese decken vor allem Verfolgung zwischen Anforderungen, aber auch deren Allokation an Systemkomponenten und die Verknüpfung mit Validierungsmaßnahmen ab. Ein

weiteres prominentes Werkzeug im Bereich der Elektronikentwicklung ist Reqtify[5], welches im Prinzip beliebige Verknüpfungen zwischen Design Artefakten erzeugen kann, solange ein entsprechender Adapter zu dem jeweiligen Entwicklungswerkzeug existiert. Natürlich existieren noch weitere Werkzeuge, auf deren detaillierte Diskussion wird an dieser Stelle jedoch verzichtet.

2.2.3 Diskussion

Anforderungen repräsentieren die projektspezifische Qualitätsdefinition. Es ist daher naheliegend, Anforderungen bei der Konzeption einer effizienten Kalibrierung der Qualitätsmessung auf ein Projekt (**Ziel 2**) zu berücksichtigen. Denn eine Aussage über Qualität und Erfüllung aller funktionalen wie nicht funktionalen Anforderungen wäre ein adäquates Qualitätsmaß in einem Projekt. Folgende Probleme lassen sich im aktuellen Stand der Technik jedoch identifizieren:

- *Hohe Heterogenität*: Vielzahl existierender Techniken zur Anforderungsdefinition mit unterschiedlichen Formalisierungsgraden (z.B. natürlich sprachlich, Boilerplates, UML basierte graphische Repräsentation, formale Automaten), die in der Regel in Kombination eingesetzt werden.
- *Funktional vs. nicht-funktional:* Die Beschreibung funktionaler Anforderungen wird oft getrennt von der Beschreibung nicht funktionaler Anforderungen betrachtet.

Anforderungsmetamodelle sind in diesem Zusammenhang ein Mittel, die verschiedenen Formalismen und Anforderungsarten auf eine einheitliche semantische Basis zu stellen. Existierende Modelle (z.B. SysML, EAST-ADL2, Darwin4Req, Modell nach Göknil) gehen jedoch selten über die Definition der typischen Anforderungsattribute und Kategorien hinaus. Selbst bei Existenz eines entsprechenden Metamodells, reicht die Definition von Anforderungen allein nicht aus, um eine projektspezifische Qualitätsdefinition zu erlangen. Sie beinhaltet ebenfalls eine eindeutige Allokation dieser Anforderungen an Bestandteile des Systems, welches diese umsetzen.

Obwohl die Einführung und systematische Umsetzung einer durchgängigen Anforderungsverfolgung in der Praxis immer noch eine Herausforderung und mit Aufwand verbunden ist, lässt sich Folgendes festhalten:

- Es existieren Konzepte und Lösungen sowohl im akademischen (z.B. TraceChange, Referenzmodelle Jarke/Ramesh, EAST-ADL) als auch im kommerziellen Bereich (z.B. Reqtify, DOORS basierte Lösungen, SysML). Ansätze (zumeist manuell) sind in mehreren Unternehmen etabliert. Dies gilt insbesondere für den Bereich sicherheitskritischer Systeme, bei denen die Nachverfolgung von Anforderungen von Sicherheitsstandards gefordert ist.

[5] http://www.geensoft.com/en/article/reqtify/

- Auch wenn sich Definition und unternehmensspezifische Implementierungen im Detail unterscheiden können, existiert doch eine relativ stabile Menge an Verknüpfungstypen, die sich in mehreren Ansätzen wiederfinden. So beschreiben Verknüpfungen Anforderungsverfeinerungen, wie Anforderungen umgesetzt und wie sie validiert/verifiziert werden.
- Die Verknüpfung von Anforderungen mit Systemkomponenten liefert eine detaillierte Definition, welche Qualitätsmaßstäbe an eine Komponente gestellt werden.

Anforderungsmetamodelle und Anforderungsverfolgung sind Grundlage zur Adressierung von **Ziel 1 - Einheitliche Qualitätsdefinition**, es existiert jedoch bisher kein Anforderungsmetamodell, welches eine gemeinsame Sicht auf alle Anforderungsarten und Formalisierungen liefert. Das Konzept der Anforderungsverfolgung ist ein elementarer Baustein zur Adressierung von **Ziel 2 – Kalibrierung auf einzelne Projekte**. Die folgende Tabelle fasst die Zielerfüllung der vorgestellten Ansätze zusammen. Es gilt dieselbe Nomenklatur wie in Abschnitt 2.1.4.

	Z1: Einheitliche Qualitätsdefinition	Z2: Effiziente Projekt-spezifische Auswertung	Z3: Integration V&V Ergebnisse	Z4: Domänenübergreifende Qualitätsbetrachtung	Z5: Entwicklungsbegleitende Auswertung	Z6: Berücksichtigung von Prozess- und Projektkontext	Z7: Automatische Datenerfassung und -integration	Z8: Anpassbarkeit an bestehende Entwicklungsprozesse
SysML		(X)						
EAST-ADL2	(X)	(X)	X	(X)				
Modell nach Göknil et. Al.		(X)						
Dawin4Req, MemVatex, ReMIAS		(X)	X	(X)				
NFR Methode		(X)						
Von Knethen								
Ramesh/Jarke		(X)						

Tabelle 2: Zielerfüllung der Ansätze zur Anforderungsdefinition

Anforderungen und deren Nachverfolgbarkeit sind jedoch nur ein Teil eines umfassenden Qualitätsmonitorings, wie es in dieser Arbeit angestrebt wird. Darüber hinaus gilt es, Analyseergebnisse aus Validierungs- und Verifikationsaktivitäten der definierten Anforderungen zu berücksichtigen und zu interpretieren. Diesem Aspekt widmet sich das nächste Kapitel. Der letzte Abschnitt bewertet abschließend die Ansätze gegen die aufgestellten Ziele.

2.3 Qualitätssicherung in der Produktentwicklung

Der dritte Abschnitt untersucht den Stand der Technik im Bereich der Qualitätssicherung in der Produktentwicklung. Es führt in das allgemeine Vorgehen zur Verifikation und Validierung in der Systementwicklung ein und legt dann einen Fokus auf Arbeiten im Bereich Qualitätsmodellierung und -bewertung.

2.3.1 Verifikation und Validierung

Ein typisches Vorgehensmodell in der Entwicklung komplexer Systeme ist das in Abbildung 15 dargestellte V-Modell. Es hat seinen Ursprung in der Entwicklung von Softwareprojekten durch öffentliche Einrichtungen der Bundesrepublik Deutschland, zuerst im militärischen und später im zivilen Bereich. Das V-Modell XT ersetzte 2005 das klassische V-Modell und liegt mittlerweile in der Version 1.3 vor (VMXT, 2009). Auch andere Entwicklungsdomänen verwenden das V-Modell. So z.B. in der Mechatronikentwicklung in Form der VDI2206 (VDI, 2004) oder in der Elektronikentwicklung, in denen das V-Modell das Vorgehen in Sicherheitsstandards (siehe z.B. (ISO26262, 2010)) prägt. Das klassische V-Modell hat gerade aus Sicht der Qualitätssicherung den klaren Nachteil, dass eine Eigenschaftsabsicherung erst zur Systemintegrationszeit geschieht. Fehler werden so zu spät entdeckt. Literatur und Praxis erweiterten daher das V-Modell, um bereits in den frühen Entwicklungsphasen, also auf der linken Seite des „V“s, Verifikations- und Validierungsmaßnahmen durchzuführen. Entwicklungsprozesse enthalten Validierungs- und Verifikationsaktivitäten auf allen Ebenen, sowohl auf der linken als auch auf der rechten Seite des „V’s“. Sie nehmen einen zentralen Abschnitt in der Produktentwicklung und am Gesamtaufwand derselben ein. Typischerweise wird das in Abbildung 15 dargestellte Vorgehen verfolgt:

Während der Produktentwicklung werden auf jeder Ebene Anforderungen definiert und in folgenden Entwicklungsschritten durch Systementwürfe adressiert. Ergebnis dieser Aktivität sind Design Modelle, welche z.B. die Architektur des Systems oder (auf Segmentebene, siehe Abbildung 15) einzelner Subsysteme darstellen. Begleitend werden regelmäßig Validierungs- und Verifikationsaktivitäten durchgeführt. Ziel dieser Aktivitäten ist der Nachweis der Anforderungsqualität und der Anforderungserfüllung. Sind diese erfolgreich nachgewiesen, werden die Anforderungen für die jeweils nächste Ebene verfeinert. Dieser Prozess wird solange wiederholt, bis die Ebene von Hard- und Software erreicht ist. Abbildung 15 enthält zwei Ebenen, je nach Unternehmen können es aber auch mehrere sein. Derselbe Prozess wiederholt sich für die Entwicklung der Hard- und Softwarekomponenten. Die rechte Seite des V-Modells besteht hauptsächlich

aus der Integration von Hardware und Software zu Teilkomponenten und deren finalen Integration in das Gesamtprodukt, um mit dem Gesamtsystem eine Produktvalidierung durchzuführen.

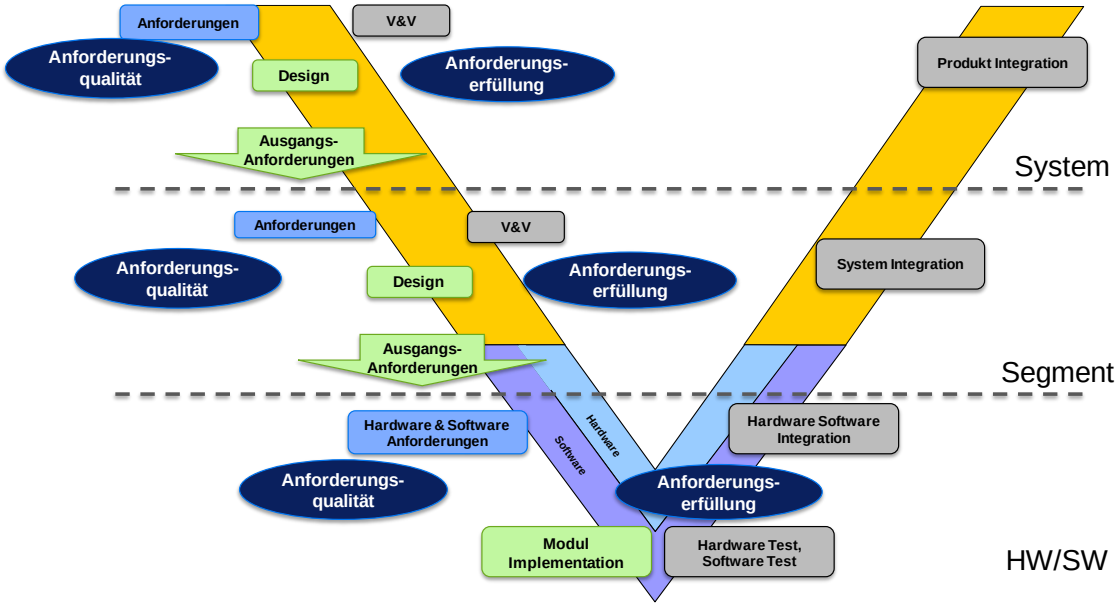

Abbildung 15: V-Modell

Es folgt eine Beschreibung was unter Anforderungsqualität und –erfüllung im Detail verstanden wird:

Anforderungsqualität

Nur bei guter Anforderungsqualität, ist die Aussage über den Anforderungserfüllungsgrad verlässlich. Existierende Arbeiten aus Forschung (Katasonov, 2006) und Standardisierung (ISOWD29148.3, 2009) unterteilen Anforderungsqualität in Charakteristika für einzelne Anforderungen und ganze Anforderungssets.

Die Anforderungsverfolgung spielt bei der Bewertung eine zentrale Rolle. Anforderungen werden über verschiedene Ebenen von Systemebene über einzelne Segmente bis hin zur Hardware und Software Ebene entlang des V-Modells verfeinert. Daher ist die Überprüfung der Konformität von Anforderungen zu den übergeordneten Ebenen zentral für die Bewertung der Anforderungsqualität. Wurden alle übergeordneten Anforderungen berücksichtigt? Bilden sie die übergeordneten Anforderungen konsistent und korrekt ab?

Anforderungsqualitätsattribute lassen sich mittels verschiedener Methoden validieren. Viele Methoden sind jedoch nur in bestimmten Entwicklungsphasen, für eine bestimmte Anforderungsrepräsentation brauchbar oder für eine bestimmte Anforderungsart nützlich. Das Qualitätsbewertungsframework muss daher eine Anpassbarkeit an den unternehmensspezifischen Anforderungsentwicklungsprozess bieten. Qualitätscharakteristika werden daher aus einer abstrakten Sicht berücksichtigt und verwenden die durch existierende Methoden oder manuelle Reviews gelieferten Ergebnisse weiter.

Definition Qualität einer einzelnen Anforderung

Eine einzelne Anforderung besitzt eine Menge an Qualitätsattributen, deren Wert entweder wahr oder falsch ist. Die folgende Tabelle beschreibt die Qualitätsattribute und deren Quelle.

Qualitätsattribut	Beschreibung	Quelle
Vollständig	Die formulierte Anforderung ist vollständig, d.h. sie benötigt keine Erweiterungen und ist Aussagekräftig genug.	(ISOWD29148.3, 2009) (ECSS-E-ST-10C, 2009) (ECSS-E-ST-40C, 2009)
Eindeutig	Jede Anforderung ist eindeutig zu definieren, so dass nur eine Art der Interpretation möglich ist.	(ISOWD29148.3, 2009) (ISODIS26262, 2009) (Katasonov, 2006)
Notwendig	Die Anforderung ist notwendig, damit die zugehörige Menge an Anforderungen vollständig ist.	(ARP4754A, 1996)
Verifizierbar	Jede Anforderung ist so zu definieren, dass die Erfüllung der Anforderung durch das System nachweisbar ist.	(ECSS-E-ST-40C, 2009) (ECSS-Q-ST-80C, 2009) (ISODIS26262, 2009)
Umsetzbar	Die Anforderung muss unter technologischen, monetären, zeitlichen und anderen Bedingungen umsetzbar sein.	(ISOWD29148.3, 2009) (ISODIS26262, 2009) (Katasonov, 2006) (ECSS-E-ST-40C, 2009)
Verfolgbar	Ist jede Anforderung zu ihrem Ursprung (Kundenanforderung, andere technische Anforderung, Design Entscheidung, Analyseergebnis) verfolgbar? Ist jede Anforderung an eine Komponente allokiert?	(ARP4754A, 1996) (ECSS-E-ST-10C, 2009) (ISOWD29148.3, 2009) (ECSS-E-ST-40C, 2009) (ECSS-Q-ST-80C, 2009) (ISODIS26262, 2009)
Atomar	Eine Anforderung ist atomar, wenn sie keine Aussagen enthält, die besser als zwei oder mehr Anforderungen beschrieben werden sollten.	(ISOWD29148.3, 2009) (ISODIS26262, 2009) (ARP4754A, 1996)

Konsistent	Eine Anforderung enthält keine Widersprüche (interne konsistenz).	(ISODIS26262, 2009)
Priorisiert	Die Anforderung muss priorisiert sein.	(Katasonov, 2006)
Umgebungsinformationen sind verfügbar	Annahmen auf denen die Anforderung basiert sind identifiziert und validiert.	(ARP4754A, 1996) (ECSS-E-ST-10C, 2009) (ECSS-E-ST-40C, 2009)

Tabelle 3: Qualitätscharakteristika einzelner Anforderungen

Hervorzuheben ist das letzte Qualitätsattribut in Tabelle 3: Dokumentation und Verfolgung des Verifikationsstatus von Annahmen unter denen die Anforderung gültig ist. Gerade in frühen Entwicklungsphasen basieren viele Aktivitäten noch auf unsicheren Anforderungen bzw. Annahmen auf denen Anforderungen basieren (z.B. Geschwindigkeit einer Hardwarekomponente), die erst in späteren Entwicklungsphasen oder vielleicht sogar durch Zulieferer validiert werden. Die Dokumentation von Annahmen und deren Validierungsstatus (wie z.B. beim Contract Based Design (Damm, 2005)) erlaubt die Berücksichtigung der Unsicherheit früher Anforderungen bei der Qualitätsbewertung.

Definition Qualität Anforderungsmodell

Die Qualität eines Anforderungsmodells berechnet sich zum einen aus den Mittelwerten der Qualitätsattribute (QA) einzelner Anforderungen. Zum anderen existieren QA, die sich auf ganze Anforderungsmodelle beziehen, die in der folgenden Tabelle genauer erläutert werden.

Qualitätsattribut	Beschreibung	Quelle
Intern konsistent	Das Anforderungsset besitzt keine zwei Anforderungen, die untereinander widersprüchlich oder redundant sind.	(ISOWD29148.3, 2009) (ISODIS26262, 2009)
Extern konsistent	Das Anforderungsset besitzt keine Anforderung, die widersprüchlich oder redundant mit Anforderungen der übergeordneten Entwicklungsphase sind (z.B. im Vergleich zu Kundenanforderungen).	(ISOWD29148.3, 2009) (ISODIS26262, 2009)

Vollständig	Vollständigkeit bedeutet, dass eine Menge an Anforderungen vollständig die Anforderung der vorherigen Ebene umsetzt. Je nach Formalisierungsgrad der Anforderungen ist dies basierend auf Anforderungsverfolgungs-konzepten oder aber auch durch semantische Überprüfungen durchführbar.	(DO178B, 1992), (ECSS-E-ST-10C, 2009) (ISOWD29148.3, 2009) (ISODIS26262, 2009)

Tabelle 4: Qualitätscharakteristika einer Menge von Anforderungen

Die genannten Charakteristika sind Bestandteil verschiedener Standards, es ist jedoch nicht zwangsläufig als finales Set festgelegt. Je nach konkreter Ausprägung des Anforderungsmanagementprozesses sind die genannten Attribute anpassbar und erweiterbar.

Anforderungserfüllung

Es ist nachzuweisen, dass die entwickelten Design Modelle, die definierten Anforderungen erfüllen. Hierfür existieren diverse Methoden, deren komplette Beschreibung den Rahmen dieser Arbeit bei weitem sprengen würde. Es folgt daher eine grobe Übersicht verschiedener Methodenarten, um die vorhandene Bandbreite zu verdeutlichen. Weit verbreitet sind manuell durchgeführte Peer Reviews, bzw. Inspections (siehe z.B. (Fagan, 1976)) in denen Designdokumente mittels eines formalen Prozesses auf Fehler untersucht werden. Weitere existierende Testmethoden sind z.B. das anforderungsbasierte Testen, in dem für jede Anforderung ein oder mehrere Testfälle definiert werden oder auch simulationsbasierte Verifikation, zum Nachweis der Korrektheit funktionaler Modelle (z.B. mit Matlab/Simulink[6]). Ein weiteres Feld umfasst statische Verifikationsmethoden wie Model Checking (siehe z.B. (Clarke, 1999)), Theorem Proving (siehe z.B. (Fitting, 1996)) oder abstrakte Interpretation (siehe z.B. (Clarke, 1994)), die jedoch im Vergleich zu den vorher genannten Methoden in der industriellen Praxis noch weniger verbreitet sind. Alle diese genannten Methoden weisen bestimmte Systemeigenschaften nach, benötigen jedoch, bis auf die Reviews, immer ein ausführbares System, bzw. eine funktionale Beschreibung.

Nun existieren jedoch auch Eigenschaften des Systems die sich eben nicht unmittelbar durch die Ausführung des Systems und der Beobachtung des Verhaltens dieses Systems bestimmen lassen. Oft verbindet man Qualität mit teilweise relativ abstrakten Systemeigenschaften wie z.B. Wiederverwendbarkeit oder Wartbarkeit. Eigenschaften die durch klassische Testverfahren nicht ohne weiteres quantifizierbar sind, jedoch ebenfalls einen elementaren Bestandteil der Systemqualität bilden. Für die Strukturierung der vielen

[6] http://www.mathworks.de/products/featured/early-verification/index.html

verschiedenen qualitätsrelevanten Eigenschaften und deren Nachweis existiert seit dem Ende der 70er Jahre des letzten Jahrhunderts der Bereich der Qualitätsmodellierung und –bewertung mit einem Fokus auf Softwareentwicklung. Dieser Abschnitt widmet sich Arbeiten dieses Themenkomplexes, welcher offensichtlich einen nahe verwandten Bereich darstellt. Die Untersuchung wird demonstrieren, dass eine Vielzahl heterogener Qualitätsmodelle existiert, die verschiedene Ziele verfolgen und unterschiedliche Aspekte von Qualität adressieren. Abschnitt 2.3.2 führt zunächst grundsätzliche Definitionen und Begriffe rund um Qualitätsmodelle ein und stellt existierende Ansätze zur Qualitätsmodellierung vor. Zur Einordnung der vorgestellten Ansätze werden anschließend mögliche Klassifikationen von Qualitätsmodellen vorgestellt. Ebenfalls von Bedeutung für diese Arbeit sind Vorgehensmodelle und Qualitätsbewertungsframeworks für die Bewertung von Produktqualität, die nicht nur die bloße Modellierung von Qualitätseigenschaften betrachten, sondern eine Anleitung zur Umsetzung entsprechender Initiativen in Unternehmen geben. Hierzu existierende Ansätze untersucht Abschnitt 2.3.3.

2.3.2 Qualitätsmodellierung

Bereits Garvin hat mit seinen fünf Sichten auf das Konzept Qualität (siehe Qualitätsdefinition in 1.2.1) und dem Satz

„Quality is a complex and multifaceted concept“(Garvin, 1984)

die Mannigfaltigkeit des Konzepts Qualität treffend beschrieben. Es verwundert daher nicht weiter, dass auf Grund der verschiedenen Sichten auf Qualität eine Vielzahl heterogener Qualitätsmodelle entstand, die sich jedoch in Definition und Anwendung unterscheiden. An dieser Stelle sei noch einmal explizit auf die Qualitätsmodell Definition aus Abschnitt 1.2.2 als eine Sammlung von Kriterien zur Bewertung eines Gegenstands (in diesem Fall das Produkt) hingewiesen. Die in (Zollondz, 2006) auch als Qualitätsmodelle bezeichneten Ansätze wie Qualitätskreis Modell oder RCPA Modell sind nicht Gegenstand der Untersuchung, da es sich hier um Vorgehensmodelle handelt, nicht jedoch um Modelle zur Strukturierung von Produktqualitätseigenschaften. Der folgende Abschnitt beschreibt zunächst die prominentesten Vertreter der Qualitätsmodellierung in chronologischer Reihenfolge und verwendet dabei der Eindeutigkeit halber stets die Definitionen aus Abschnitt 1.2.1:

Klassische Ansätze

Nahezu alle im Laufe der Zeit entwickelten Qualitätsmodelle ähneln in ihrer Art der Strukturierung den beiden historischen Softwarequalitätsmodelle nach Boehm (Boehm, 1976), (Boehm, 1978) und McCall (McCall, 1977). Boehm und McCall formulieren Hauptcharakteristika der Qualität von Software Produkten aus einer Benutzerperspektive. Die abstrakten Charakteristika sind in den meisten Fällen qualitätsrelevante externe Produktcharakteristika und werden hierarchisch bis hin zu messbaren Produkteigenschaften (auch als interne Produktcharakteristika bezeichnet) weiter verfeinert. Diese Methode wird oft auch als Factor-Criteria-Metric (FCM) bezeichnet. Ein Ausschnitt der

Qualitätsmodelle nach Boehm und McCall dargestellt in Abbildung 16 visualisiert das Vorgehen.

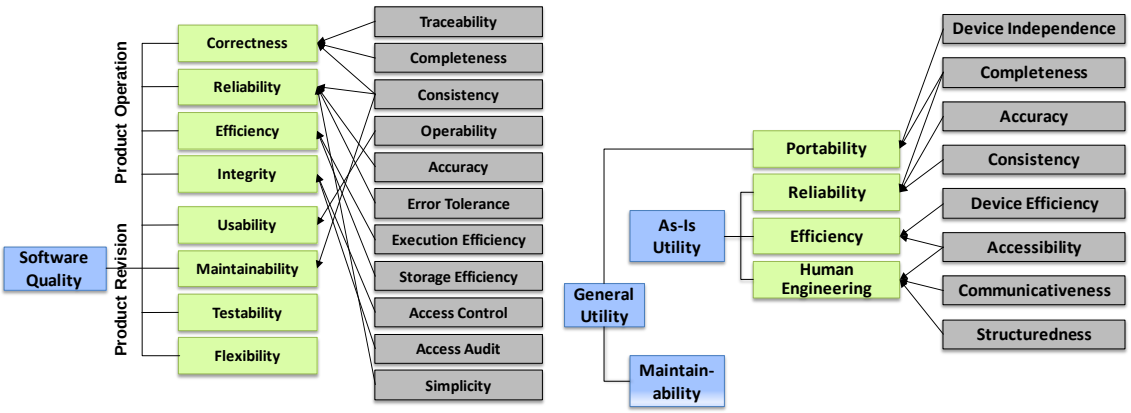

Abbildung 16: Ausschnitt Qualitätsmodell nach McCall (links) und Boehm (rechts)

Die folgenden Abschnitte zeigen, dass das Vorgehen der hierarchischen Dekomposition in vielen Ansätzen ebenfalls verfolgt wird. Ein Kritikpunkt an die Modelle von McCall und Boehm ist die festgelegte Anzahl von maximal drei Ebenen (siehe z.B. (Deißenböck, 2007)), die oft nicht ausreichen um komplexere Qualitätscharakteristika bis auf messbare Qualitätsattribute herunter zu brechen. Die Begrenzung wird bei anderen Qualitätsmodellen daher oft nicht eingehalten.

Weitere frühe Qualitätsmodelle entstanden durch Kitchenham (Kitchenham, 1987), Deutsch (Deutsch, 1988), Nance (Nance, 1992) und Bowen (Bowen, 1976). Diese Lösungen haben jedoch keine breite Akzeptanz in der Praxis gefunden, da sie keine Ansätze zur Operationalisierung und Anpassung auf den Projektkontext bieten. Selbiges gilt zwar auch für die Ansätze nach McCall und Boehm, diese hatten jedoch Pioniercharakter.

ISO Qualitätsstandards

Basierend auf den ersten Modellen nach Boehm und McCall Ende der 70er Jahre entstanden weitere Arbeiten zur Definition von Softwarequalitätsmodellen. Hierzu gehört der ISO/IEC 9126 Standard, ein von Boehm inspiriertes Modell für Software Qualität, welches aus 6 Hauptcharakteristika und 27 Subcharakteristika besteht. Es handelt sich um einen generischen Standard, für den es Maßnahmen zur Anpassung auf das betreffende Unternehmen bzw. Projekt bedarf. Der Standard besteht aus folgenden vier Teilen:

- *Quality Model:* Beschreibt das Quality Model Framework und identifiziert die für Software Produkte relevanten Charakteristika und Sub-Charakteristika. (ISO9126-1, 2001)
- *External metrics:* Beschreibt die externen Metriken, die verwendet werden, um die definierten Charakteristika und Sub-Charakteristika zu messen. (ISO9126-2, 2002)

- *Internal metrics:* Beschreibt die internen Metriken, die verwendet werden, um die definierten Charakteristika und Sub-Charakteristika zu messen. (ISO9126-3, 2002)
- *Quality in use metrics:* Beschreibt die Metriken um die Effekte der kombinierten Qualitätscharakteristika für den Benutzer zu messen. (ISO9126-4, 2002)

Die ersten drei Teile beschäftigen sich demnach mit der Messung der Softwarequalität während der Entwicklung, der vierte Teil widmet sich der Qualitätsbewertung durch den Benutzer.

Der Standard wurde das erste Mal 1991 veröffentlicht und erfuhr 2001 eine Überarbeitung, da sich die erste Version des Standards verschiedenen Kritiken ausgesetzt sah. So wurde er z.B. als nicht umfassend genug betrachtet, als dass es für alle Projekte gültig wäre (Sommerville, 2004), (Wallace, 2001). Außerdem sei er schwierig zu verstehen (Sommerville, 2004), (Kitchenham, 1997). Sowohl Dromey (Dromey, 1995) als auch Bøegh (Bøegh, 1999) bezeichnen den Standard als zu vage, um Softwarequalität wirklich zu messen. Der Standard stellt einen deskriptiven Ansatz dar und ist allein nicht dazu geeignet, um Messungen durchzuführen.

Weitere Studien versehen auch den überarbeiteten Standard mit ähnlich lautenden Kritiken. Kitchenham et al. stufen den Standard in (Kitchenham, 2005) als fehlgeschlagen ein, da der Standard die selbst gesteckten Ziele nicht erfüllt. Die genannten Probleme in der durchgeführten Studie lassen sich wie folgt zusammenfassen:

- Es fehlen Anweisungen, wie die einzelnen messbaren Werte zu Subcharakteristika bzw. Charakteristika aggregiert werden müssen. Diese Entscheidung ist einzelnen Unternehmen überlassen.
- Nach Meinung der Autoren fehlen einige wichtige Charakteristika bzw. sind zu abstrakt, was die These dieser Arbeit untermauert, dass es kein einheitliches Qualitätsmodell gibt. Meinungen über wichtige Eigenschaften bzw. die Qualität eines (Software-) Produkts sind subjektiv und gehen auseinander.
- Es existieren widersprüchliche Konzepte und Metriken.

Als Nachfolger des ISO/IEC 9126 Standards für Software Qualität und des ISO 14598, welcher ein methodisches Vorgehen zur Etablierung der Evaluation von Software Qualität beschreibt (siehe auch Abschnitt 2.3.3 dieser Arbeit), ist im Jahr 2005 der ISO/IEC 25000 Standard (ISO25000, 2005) veröffentlicht worden, Teile der Serie erschienen erst 2007.

Dromey

Ein weiterer Ansatz zur Qualitätsmodellierung wurde von Dromey in (Dromey, 1995) vorgestellt. Dromey kritisiert die oft fehlende Berücksichtigung von Produktcharakteristika, sowie die fehlende Verbindung zwischen Produktcharakteristika und Qualitätsattributen. Die Qualitätsattribute sind nicht direkt messbar und lassen sich daher auch

nicht direkt bei der Entwicklung berücksichtigen. Dies ist nur indirekt über Produktcharakteristika möglich. Übertragen auf die Begriffswelt des ISO/IEC 9126 Standards: Dromey legt den Fokus auf interne Produktcharakteristika und deren Verbindung zu externen Produktcharakteristika und betrachtet dabei den Code bzw. die Implementierung der Software.

In (Dromey, 1996) stellt Dromey Qualitätsmodelle für unterschiedliche Phasen des Softwareentwicklungsprozesses vor. Hierzu gehört ein Qualitätsmodell für Anforderungen, Entwurf und Implementierung. Letzteres ist an das Modell nach ISO/IEC 9126 Standard angelehnt, welches ausschließlich die Implementierung betrachtet. Die Entwicklung mehrerer Qualitätsmodelle für die unterschiedlichen Entwicklungsphasen ist ein signifikanter Unterschied zum ISO Standard.

Dromey stellt fünf Richtlinien für die Entwicklung von Qualitätsmodellen auf (Dromey, 1995), mit deren Hilfe er drei Qualitätsmodelle für Anforderungen, Entwurf und Implementierung erstellt:

- Identifiziere die abstrakten Qualitätsattribute eines Produkts.
- Identifiziere die Komponenten des Produkts.
- Identifiziere und klassifiziere die signifikantesten und greifbarsten qualitätsbeeinflussenden Aspekte für jede Komponente.
- Führe mehrere Axiome ein oder verbinde die Produktcharakteristika mit Qualitätsattributen.
- Bewerte das Modell, identifiziere Schwachstellen, verbessere oder verwerfe es und starte neu.

Insgesamt lässt sich das Qualitätsmodell nach Dromey als eine Erweiterung des ISO Standards auf mehrere Entwicklungsphasen bezeichnen, welches detaillierter in seiner Ausprägung ist. Qualitätsattribute bzw. externe Qualitätseigenschaften werden deutlich weiter herunter gebrochen als im ISO Standard.

Weitere „klassische“ Qualitätsmodelle

Es existieren eine Reihe weiterer Qualitätsmodelle für Software, die auf Grund ihrer ähnlichen Art der Modellierung nicht alle detailliert beschrieben werden. Hierzu gehört das Qualitätsmodell von SATC (Hyatt, 1996), welches dieselbe Strukturierungsmethodik wie der ISO 9126 Standard verwendet, jedoch externe Qualitätsparameter für verschiedene Schritte im Entwicklungsprozess entwickelt (Requirements Quality, Product(Code) Quality, Implementation Effectivity, Testing Effectivity). FURPS (Grady, 1987) formuliert die Qualitätskriterien Funktionalität, Benutzbarkeit, Zuverlässigkeit, Effizienz und Änderbarkeit, welche ebenfalls Bestandteil der ISO 9126 sind. Das von der MITRE Corporation entwickelte Software Quality Assessment Exercise (kurz SQAE) (Martin, 1996) beschreibt ein Qualitätsmodell bestehend aus vier festen Qualitätsbereichen und sieben Qualitätsfaktoren, die jeweils Einfluss auf ein oder mehrere

Qualitätsbereiche haben. Das definierte Qualitätsmodell ist fest und sieht keine Spezialisierung auf einzelne Projekte vor. Die Vielzahl an existierenden Qualitätsmodellen im Softwarebereich zeigen, dass in der Domäne kein allgemeingültiges Modell existiert. Es gibt zwar oft Überschneidungen, aber auch immer wieder Unterschiede bei verschiedenen Charakteristika.

Aktivitätsbasierte Qualitätsmodelle

Ein weiteres Modell zur Beschreibung von Softwarequalität ist im Rahmen von Arbeiten des Software & Systems Engineering Lehrstuhls der Technischen Universität München entstanden. Der Fachbereich beschäftigt sich mit verschiedenen Facetten der Qualitätsbewertung von Softwaresystemen in dessen Rahmen ein aktivitätsbasiertes Metamodell zur Definition von spezifischen Qualitätsmodellen entstanden ist.

Wie Broy et al. (Broy, 2005) und Wagner und Deißenböck in (Wagner, 2007a) erläutern, haben auch sie das Ziel einen holistischen Ansatz zur Bestimmung von Softwarequalität zu entwickeln. Es existieren eine Vielzahl an Softwarequalitätsmodellen, die aber isoliert existierende Lösungen sind, die keine integrierte Auswertung ermöglichen. Abhängigkeiten und Inkonsistenzen sind nicht erkennbar. Auf Grund dessen wurde im genannten Artikel ein Metamodell vorgestellt, welches eine einheitliche Art der Modellierung von Situations- und applikationsspezifischen Qualitätsmodellen ermöglicht, welches die Autoren für die beiden Qualitätseigenschaften Nutzbarkeit (Winter, 2007) und Wartbarkeit an (Broy, 2006), (Deißenböck, 2007) konkretisieren.

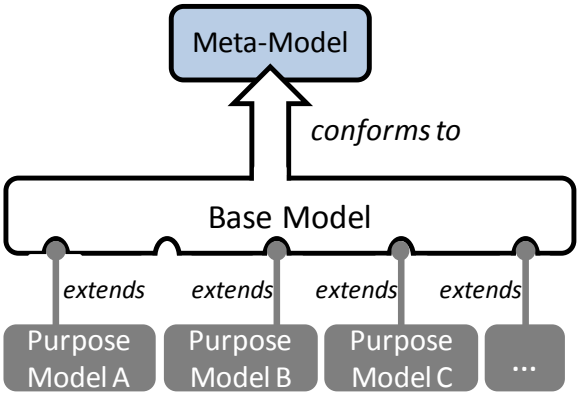

Abbildung 17: Qualitätsmodellierungsansatz nach Wagner (Wagner, 2007a)

Der in Abbildung 17 dargestellte Ansatz zur Qualitätsmodellierung umfasst drei Ebenen. Das *Meta-Model*, welches die allgemeine Struktur liefert, die jedes weitere Qualitätsmodell einzuhalten hat. Die Autoren geben ebenfalls ein Beispiel für ein Metamodell, betonen aber die Unabhängigkeit des Ansatzes vom konkreten Metamodell. Wichtig ist nur, dass die relevanten Eigenschaften des Systems, des Prozesses und der Umgebung inklusive deren Verbindungen beschrieben sind. Das von den Autoren vorgeschlagene Modell basiert ebenfalls auf der bereits beschriebenen Idee, externe Produktcharakteristika wie Wartbarkeit auf fassbare interne Produktcharakteristika herunter zu brechen. Die Autoren legen dabei einen Fokus auf kostenrelevante Aktivitäten, welches

sie zur Dekomposition des Konzepts Qualität verwenden. Nach Garvin (Garvin, 1984) beschreiben sie die Lösung daher als ein wertbasiertes Qualitätsmodell. Den Fokus legen sie auf Kosten, da Kosten der einzige Wert ist, in den alles konvertierbar ist. Darüber hinaus ist die Generierung von monetärem Wert das Ziel eines jeden kommerziellen Softwareprojekts. Verschiedene Stakeholder erzeugen dabei Kosten auf unterschiedliche Weise. So erzeugen z.B. Entwicklungs- und Wartungsaktivitäten während des Betriebs Kosten, die es zu minimieren gilt. Das vorgestellte Metamodell besteht daher aus zwei Baumstrukturen, auf der einen Seite die klassischen Systemeigenschaften und auf der anderen Seite die Kostenfaktoren. Abbildung 18 zeigt eine beispielhafte Implementierung des Metamodells für die Eigenschaft Wartbarkeit.

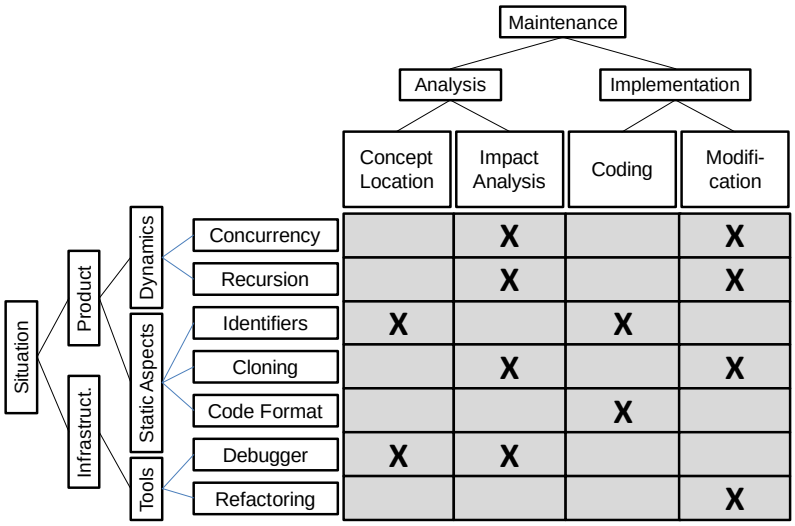

Abbildung 18: Implementierung am Beispiel Wartbarkeit (Wagner, 2007a)

Das Base-Model verwendet das Metamodell, beschreibt die allgemein relevanten Qualitätsaspekte des betrachteten Systems und stellt die Einflüsse zwischen diesen dar. Ein Base Model deckt dabei zwei Ebenen ab. Zum einen den abstrakten Teil, welcher die domänenübergreifenden wieder verwendbaren Teile des Modells enthält, sowie einen detaillierten Teil mit produktspezifischen Elementen. Die Purpose Models sind spezielle Modelle, die einem bestimmten Zweck dienen und Aussagen bzw. Vorhersagen über bestimmte Eigenschaften (z.B. Zuverlässigkeit) treffen.

Klassifikation von Qualitätsmodellen nach Tian

Die bisher beschriebenen Qualitätsmodelle lassen sich im Wesentlichen anhand von Eigenschaften und Attributen des Systems klassifizieren. Dies ist aber keinesfalls die einzige und wahrscheinlich auch keine ausreichende Klassifikation. Tian (Tian, 2004) klassifiziert Qualitätsmodelle nach ihrem Abstraktionsgrad und unterscheidet zwischen allgemeinen, produktspezifischen und sogar projektspezifischen Qualitätsmodellen.

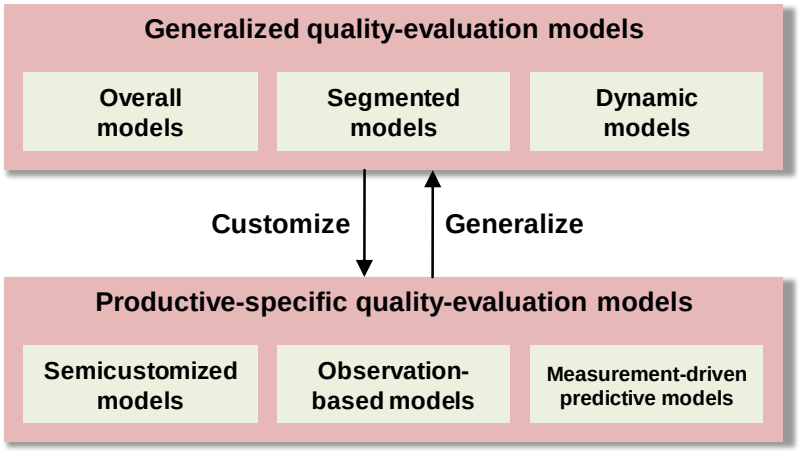

Abbildung 19: Klassifizierung von Qualitätsmodellen nach Tian (Tian, 2004)

Abbildung 19 stellt diese beiden Kategorien und deren Unterkategorien dar. Als „Generalized quality-evaluation models“ bezeichnet Tian Modelle, die Abschätzungen über industrielle Durchschnittswerte oder allgemeine Trends treffen ohne dabei eine Vielzahl projektspezifischer Daten zu benötigen. Die Unterkategorien sind folgende:

- *Overall models:* Liefert eine einzelne Zahl über die Gesamtproduktqualität.
- *Segemented models:* Liefert Abschätzungen für verschiedene industrielle Produktsegmente. Ein Beispiel ist hier die Definition der maximalen Fehler pro Stunde. Für sicherheitskritische Applikationen gelten andere Zielwerte als bei Unterhaltungssoftware.
- *Dynamic models:* Liefert den Trend der Qualitätsentwicklung über die Zeit und über mehrere Entwicklungsphasen. Anmerkung: Obwohl durch Tian als ein generalisiertes Qualitätsmodell klassifiziert, wird dieses Charakteristikum als orthogonales Konzept betrachtet. Ob nun allgemeines oder produktspezifisches Qualitätsmodell, beide können prinzipiell auch die Qualitätsentwicklung über die Zeit berücksichtigen.

„Product-specific quality-evaluation models“ werden von Tian als Modelle bezeichnet, die produkt-spezifische Daten bei der Bewertung berücksichtigen.

- *Semicustomized models:* Verwenden allgemeine Charakteristika und historische Daten über Produkt, Projekt und Umgebung, um die aktuelle Qualität zu bestimmen.
- *Observation-based models:* Verwenden Beobachtungen des Verhaltens des Softwaresystems des aktuellen Projekts, um Aussagen über die Produktqualität zu treffen. Beispiele hierfür sind Software Reliability Growth Modelle (SRGM).
- *Measurement-driven predictive models:* Verbinden Messungen mit verschiedenen Qualitätseigenschaften, um so Aussagen über diese Eigenschaften zu treffen. Diese Modelle verbinden interne Produktcharakteristika mit externen Produktcharakteristika.

Qualitätsmodelldimensionen nach Wagner und Deißenböck

Eine Klassifikation gemäß Attributen oder Abstraktionen ist immer noch keine umfassende Klassifikation von Qualitätsmodellen. Da anscheinend eine Dimension zur Klassifikation nicht ausreicht, stellen Wagner und Deißenböck (Wagner, 2007a) eine multidimensionale Klassifikation von Qualitätsmodellen vor. Wagner und Deißenböck identifizieren die folgenden sechs Qualitätsdimensionen für Softwarequalitätsmodelle:

1. Zweck Im Allgemeinen verfolgt der Einsatz von Qualitätsmodellen ein oder mehrere der folgenden Zwecke:

a. *Konstruktiv:* Konstruktive Modelle beschreiben die Beziehung zwischen Aktionen und Aspekten von Softwarequalität. So beeinflusst z.B. die Verwendung bestimmter Konstrukte einer Programmiersprache die Qualität des Systems. Die Beschreibung dieser Beziehungen kann grob granular sein, erhöhen aber trotzdem das Verständnis und unterstützen bei Entscheidungen während der Entwicklung.

b. *Vorhersagend:* Vorhersagende Modelle helfen bei der Planung der weiteren Entwicklung bezüglich bestimmter Qualitätsaspekte und dienen zur Planung der Qualitätssicherung.

c. *Bewertend:* Bewertende Modelle geben eine Aussage über den aktuellen Qualitätsstand des Produkts und dienen zur Überwachung der Qualitätssicherung.

2. Sicht Qualitätsmodelle unterstützen verschiedene Sichten, deren Einteilung auf Garvin (Garvin, 1984) basiert (siehe Abschnitt 1.2.1). Lediglich die metaphysische (engl. transcendental) Sicht lassen sie außer Acht, welche sie als Basis für ein Qualitätsmodell nicht als passend erachten.

3. Attribute Eine dominante Art der Klassifikation von Qualitätsmodellen sind die Systemattribute (Zuverlässigkeit, Wartbarkeit, etc.), die sie beschreiben. Sie findet sich in Software Qualitätsstandards wie der ISO9126 wieder.

4. Phase Qualitätsmodelle konzentrieren sich auf bestimmte Phasen des Entwicklungsprozesses. Für dieselbe Eigenschaft können dabei je nach Entwicklungsphase unterschiedlich abstrakte Modelle zum Einsatz kommen. Mit jeweils unterschiedlicher Aussagekraft.

5. Technik Qualitätsmodelle fokussieren sich oft auf bestimmte Techniken der Fehlererkennung. So existieren verschiedene Modelle, die speziell für Code Inspektionen entwickelt wurden und Ergebnisse nutzen, die nur durch diese Technik geliefert werden.

6. Abstraktion Die letzte Dimension beschreibt den Abstraktionsgrad des Qualitätsmodells. Wagner und Deißenböck zitieren an dieser Stelle Tian (Tian, 2004), welcher allgemeine und produktspezifische Qualitätsmodelle unterscheidet. Qualitätsmodelle sind entweder allgemeingültig, dafür aber relativ abstrakt, oder aber auf ein konkretes Projekt angepasst.

Im Gegensatz zu den ersten beiden Definitionen ist die Klassifikation von Wagner und Deißenböck am umfassendsten und für diese Arbeit maßgebend. Entsprechend den Anforderungen wird eine bestimmte Qualitätsdefinition angestrebt.

2.3.3 Qualitätsbewertungsframeworks

Qualitätsmodelle sind nur ein Bestandteil für die Durchführung einer quantitativen Qualitätsmessung. Da das Ziel dieser Arbeit nicht nur in der Definition von Qualität liegt, widmet sich der folgende Absatz daher existierenden Qualitätsbewertungsframeworks und Vorgehensmodellen, die Unternehmen systematisch beim Aufsetzen einer entwicklungsbegleitenden Qualitätsbewertung von der Definition über die Datensammlung bis hin zur Anpassung auf einzelne Projekte systematisch unterstützt.

ISO 14598

Ergänzend zum ISO/IEC 9126 Standard existiert ein weiterer ISO Standard, die ISO14598 (ISO14598, 1999). Dieser widmet sich der gesamten Methodik zur Messung und Bewertung von Software Qualität unter Einbeziehung des ISO/IEC 9126 Standards. Zu diesem Zweck stellt die ISO 14598 einen allgemeinen Bewertungsprozess von Software Qualität vor. Die folgende Abbildung gibt einen Überblick des Prozesses.

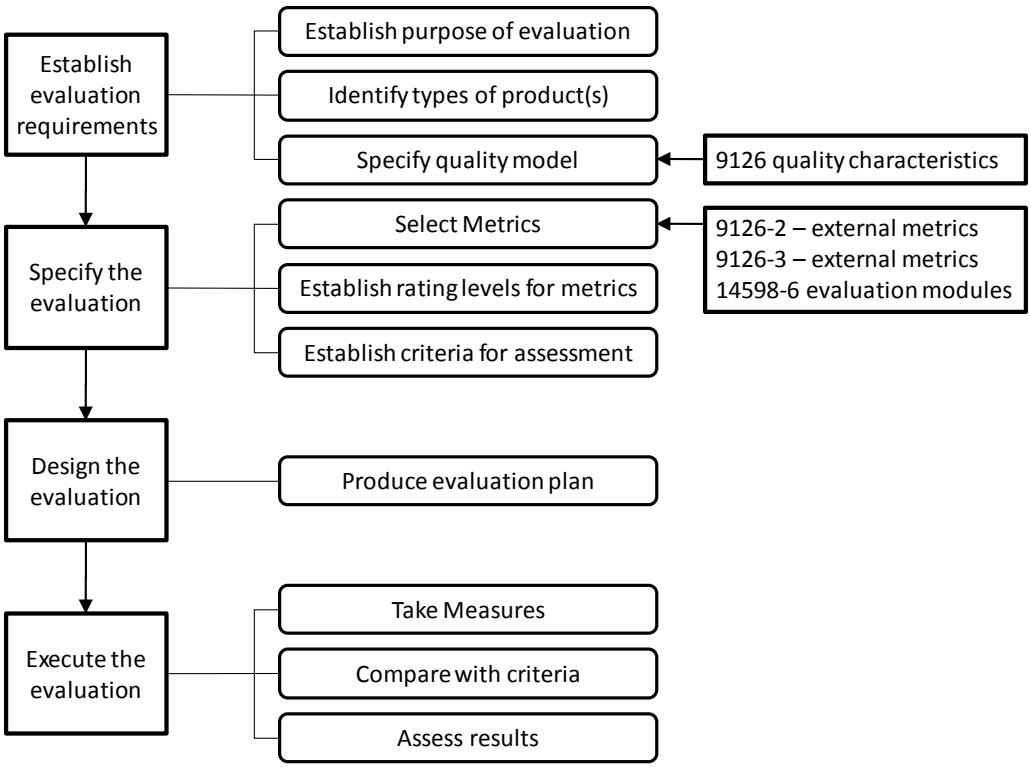

Abbildung 20: ISO 14598 Evaluierungsprozess

Eine detaillierte Beschreibung findet sich in (ISO14598, 1999). Der Standard gibt eine grobe Anleitung zum Ablauf von Messprogrammen und ist eng mit dem ISO/IEC 9126 Softwarequalitätsstandard verbunden.

Der ISO 14598 bietet zwar die Freiheit ein speziell angepasstes Qualitätsmodell zu verwenden, gibt aber keine Anleitung, wie dies durchzuführen ist, noch wie man es in Entwicklungsprozesse integriert. Das Prozessmodell bietet demnach bestenfalls eine grobe Richtlinie für die Entwicklung des Vorgehensmodells in Kapitel 3, kann aber keine Ziele dieser Arbeit erfüllen.

EMISQ

Plösch et al. stellen in (Plösch, 2007) eine Methodik zur systematischen Bewertung interner Softwareproduktcharakteristika namens EMISQ (kurz für Evaluation Method for Internal Software Quality) vor. Das Modell ist eine Erweiterung des bereits beschriebenen ISO-14598 Standards.

Die Methodik besteht dabei aus 8 Hauptaktivitäten, die wiederum in weitere Subaktivitäten unterteilt sind. Die ersten vier Schritte bis „Select Metrics“ sind dabei identisch mit dem ISO-14598 Standard. Die folgenden beiden Schritte des ISO Standards entfallen jedoch bei der EMISQ Methodik. Automatisch berechnete Metrik Ergebnisse werden zusätzlich manuell durch einen Experten bewertet. Nur dieser besitzt laut den Autoren die Kompetenz den Kontext des Produkts bzw. der Software mit in die Bewertung einzubeziehen. Die EMISQ Methode beschränkt sich auf die Untersuchung des Quellcodes und verwendet an dieser Stelle ein Qualitätsmodell, welches auf dem ISO 9126 Standard basiert und dieses mit Elementen von SATC (Hyatt, 1996), FURPS (Grady, 1987) und des Qualitätsmodells des Software Engineering Institutes (SEI) anreichert. Jeder Wert einer Metrik innerhalb des Qualitätsmodells wird durch einen Experten als ok, kritisch oder sehr kritisch eingestuft. Ein Qualitätsattribut wird durch Medianbildung über allen Subattributen berechnet, wobei der Experte auch hier die Möglichkeit hat, den resultierenden Median zu überschreiben. Weiterhin erhält jedes der Subattribute ein Gewicht. Die Gesamtqualität wird ebenfalls über den Median aller Qualitätsattribute gebildet.

Die Autoren merken an, dass die vorgestellte Methodik auf Grund des manuellen Aufwands für die entwicklungsbegleitende Bewertung von Software nicht geeignet ist, weshalb in (Plösch, 2008) ein Werkzeug zur Unterstützung der EMISQ Methode vorgestellt wird. Das Werkzeug unterstützt jedoch lediglich die Evaluierung von Softwarecode, nicht jedoch die Evaluierung von anderen Ergebnissen innerhalb des Entwicklungsprozesses und ist daher nicht geeignet für domänen- und Abstraktionsebenen übergreifende Entwicklungsvorhaben. Lediglich die eher Werkzeug orientierten **Ziele 5 und 6** sind erfüllt.

EQM

In (Paquin, 2000) stellen Paquin et al. einen generischen Ansatz zur Überwachung und Bewertung der Qualität eines Projektendergebnisses vor. Die Earned Quality Method (kurz EQM) dient zur Unterstützung von Projektmanagern. Die Autoren formulieren dabei vier essentielle Anforderungen, die eine Methodik zur Bewertung und Überwa-

chung von Qualität zu erfüllen hat. Diese Anforderungen finden sich in den im Folgenden beschriebenen Schritten der EQM Methodik wieder:

Definition der Bedürfnisse des Kunden: In diesem Schritt werden die Ziele bezüglich der Qualität erfasst und auf messbare Charakteristika herunter gebrochen. Dies entspricht, obwohl in dem Artikel nicht explizit genannt, der gängigen Praxis aus der Qualitätsmodellierung externe Produktcharakteristika durch interne Charakteristika abzubilden und ähnelt einer typischen Anforderungserhebungsaktivität. Die entstehende Hierarchie wird als Quality Breakdown Structure (QBS) bezeichnet.

Bewertung und Aggregation der Präferenzen des Kunden: Jedes messbare Kriterium fließt entsprechend der Präferenzen des Kunden gewichtet in die Gesamtbewertung ein. Ferner erhält jedes Kriterium in Absprache mit dem Kunden eine Bewertungsfunktion (eine sogenannte „Value Function"). Für einige Kriterien sind Übererfüllung und Toleranzgrenzen zu berücksichtigen, für andere nicht. Der Grad der Erfüllung lässt sich durch Einsetzen von abgeschätzten Werten in die „Value Functions" berechnen. Diese Abschätzungen werden im Laufe des Projekts dabei immer konkreter. Die Gesamtqualität berechnet sich dann über eine gewichtete Mittelwertbildung über alle bewerteten Kriterien.

Bewertung der gewonnenen Qualität: In diesem Schritt werden Projektaktivitäten einer definierten Work Breakdown Structure (WBS) festgelegt, die Charakteristika der QBS beeinflussen. Hiermit geht die Methode über die reine Bewertung der Qualität hinaus. Ziel ist die Identifikation kritischer Aktivitäten und von Personen, die für (Nicht-) Veränderungen der Qualität verantwortlich sind. Für eine detaillierte Beschreibung dieses Schrittes sei auf (Paquin, 2000) verwiesen.

Bewertung von Qualitätsabweichungen und Initiierung korrigierender Aktionen: Nach Bewertung der gewonnenen Qualität wird diese mit der geplanten Qualität abgeglichen und so eine mögliche Qualitätsabweichung berechnet. Auf Basis überschrittener Grenzwerte einzelner Ziele hat der Projektmanager die Möglichkeit korrigierende Maßnahmen einzuleiten.

Ziel von EQM ist eine entwicklungsbegleitende Qualitätsüberwachung von Entwicklungsprojekten. Ein Ziel, dass sich auf dieser allgemeinen Ebene mit dem der Arbeit deckt. Wie diese Arbeit strebt EQM eine regelmäßige Qualitätsbewertung an und stellt ein Vorgehen zur projektspezifischen Erfassung einer Qualitätsdefinition vor (**Ziel 2**). Ferner allokiert die Arbeit einzelne Qualitätscharakteristika an einzelne Elemente der WBS eines Projekts und schafft so eine Integration mit dem Entwicklungsprozess (**Ziel 6**). Die Art und Weise ist jedoch aus den folgenden Gründen nicht ausreichend: Viele der genannten Interaktionen mit dem Kunden sind typische Aktivitäten der Anforderungsdefinition und sollten an dieser Stelle abgegriffen, bzw. integriert werden, anstatt als Parallelaktivität zu laufen. Sowohl die Definition der Qualitätsziele als auch deren Verknüpfung mit der WBS erzeugt zwar immer eine projektspezifische Definition, ist jedoch so speziell, dass sie in keinem anderen Projekt wiederverwendbar ist. Dies lässt die Methodik sehr aufwendig erscheinen. Es fehlen geeignete Konzepte, welche die

Wiederverwendung von Qualitätsdefinitionen und deren Kalibrierung auf einen Projektkontext erlauben. Schlussendlich beinhaltet der EQM Ansatz keine Ansätze zur automatisierten Datenerfassung und –auswertung. Der Ansatz lässt es offen, wie z.B. die in der Elektronikentwicklung benötigten V&V Ergebnisse zusammengeführt und möglichst automatisch gegen die definierten Ziele ausgewertet werden.

SQUID

SQUID ist sowohl der Name einer Methodik als auch eines zugehörigen Werkzeugs für die Qualitätssicherung und –kontrolle von Software Produkten (Boegh, 1999). Ziel des im Rahmen des Esprit Programms von der europäischen Union geförderten Projekts war die Unterstützung von Softwareentwicklungsunternehmen bei der entwicklungsbegleitenden Überwachung der Qualität ihrer Softwareprodukte. Eine Hauptanforderung war, dass Qualität kontextbezogen innerhalb eines Unternehmens und für bestimmte Projekte zu definieren ist, weshalb allgemeine Qualitätsmodelle und –standards nur eine bedingte Hilfe sind. Analog zu dieser Arbeit war es in dem Projekt daher das Ziel eine Methodik zur Qualitätsmodellierung zu entwickeln, die dieser Anforderung genügt. Sowohl die in SQUID verwendete Qualitätsdefinition als auch die verwendete Art der Qualitätsmodellierung Top-Down durch Verfeinerung externer hin zu interner Produktcharakteristika ist dabei konsistent zur Qualitätsanforderung des ISO/IEC 9126 Standards.

Innerhalb des SQUID Ansatzes werden Anforderungen an die Produktqualität durch die Festlegung von Zielwerten für externe Qualitätscharakteristika definiert. Diese Ziele werden während der Entwicklung verfolgt, indem Ziele für interne Produktcharakteristika wie z.B. Lines of Code oder Testabdeckung formuliert werden. Um schließlich die Verbindung zwischen internen und externen Qualitätscharakteristika zu unterstützen, werden Erfahrungen ähnlicher Projekte verwendet.

Neben der eigentlichen Qualitätsmodellierung werden in SQUID die projektspezifischen Objekte von Unternehmen definiert. Hierzu gehören z.B. die zu liefernden Ergebnisse, Meilensteine und die verschiedenen Software Module. Die Qualitätswerte werden von einem definierten Ergebnis an einem bestimmten Meilenstein erfasst und entsprechend bewertet. Darüber hinaus wird berücksichtigt, dass innerhalb von Softwareprodukten unterschiedliche Arten von Komponenten (Schnittstellen, Datenbankkomponenten) unterschiedliche Qualitätsanforderungen haben und daher einer entsprechenden Zuordnung bedürfen. Innerhalb des SQUID Datenmodells existieren verschiedene Arten von Messwerten: (a) ein Ist-Wert, welcher zur Entwicklungszeit erfasst wird (b) ein Zielwert, welcher definiert wird; (c) und ein abgeschätzter Wert, welcher aus vergangenen Erfahrungen oder mit Hilfe von Vorhersagemethoden bestimmt wird.

Der Prozess zur Definition und Verfolgung unternehmens- und projektspezifischer Qualitätsziele unterteilt sich in SQUID folgendermaßen:

Quality Specification: In der ersten Phase definiert ein Unternehmen, welches Qualitätsmodell zu verwenden ist. Dies kann sowohl ein internationaler Standard als auch eine unternehmensspezifische Lösung sein. Darauf folgt eine Partitionierung des Pro-

dukts in unterschiedliche Arten von Komponenten (Datenbankkomponenten, Schnittstellen, etc.) und die spezifischen Ausprägungen von Qualitätsanforderungen mit definierten Zielwerten werden diesen Arten von Komponenten zugeordnet.

Quality Planning: In der zweiten Phase werden entsprechend der definierten Qualitätsanforderungen die passenden Entwicklungsprozesse ausgewählt und Zielwerte für die internen Produktcharakteristika definiert. Dem Benutzer werden geeignete Zielwerte aus der Erfahrung mit vergangenen Projekten vorgeschlagen. Die Werte werden im Datenmodell den zu liefernden Ergebnissen zugeordnet.

Quality Control: Die Daten für die internen Produktcharakteristika werden an den definierten Meilensteinen erfasst und mit den Zielwerten verglichen. Es wird jedes interne Produktattribut gemessen und verglichen.

Quality Evaluation: In der letzten Phase des SQUID Qualitätsprozesses werden die Ist-Werte der externen Produktcharakteristika erfasst, mit den Zielwerten verglichen und auf eventuelle Qualitätsprobleme aufmerksam gemacht.

SQUID adressiert einige in Kapitel 1 definierte Ziele. So ist zumindest eine Zuordnung von Qualitätsanforderungen zu Arten von Komponenten möglich. Ferner werden entsprechend von Unternehmens- und Projektzielen die Qualitätsmodelle mit entsprechenden Zielwerten justiert, was ein gewisses Maß an Flexibilität ermöglicht. Obwohl die Definition von Zielwerten, wie im Ansatz dargestellt der Definition von Anforderungen gleich kommt, wird die Verknüpfung zum Requirements Engineering nicht deutlich, die definierten Qualitätsanforderungen werden getrennt von klassischen funktionalen Anforderungen betrachtet. **Ziel 1** aber auch **Ziel 2** sind daher nicht vollständig erfüllt. Ferner wird kein Konzept zur automatischen entwicklungsbegleitenden Erfassung der notwendigen Daten (**Ziel 5 und 7**) unterschiedlicher Produktmodelle und Analyseergebnisse vorgestellt, um in regelmäßigen Abständen und nicht erst an definierten Meilensteinen eine Bewertung der Produktqualität durchzuführen. Ebenfalls fehlt eine Integration mit dem Entwicklungsprozess (**Ziel 6**), um eine Operationalisierung der Qualitätsmessungen zu erreichen und basierend auf den Ergebnissen Projekt und Prozess zu steuern. Zweifellos ist dies durch die Berücksichtigung von Meilensteinen angedeutet, um aber im Detail zu prüfen, welche Aktivitäten durchgeführt sind und welche nicht, fehlt es SQUID an methodischer Unterstützung. Final adressiert auch SQUID „nur“ die Softwareentwicklung.

conQAT

Basierend auf den in Abschnitt 2.3.2 vorgestellten Arbeiten zur Qualitätsmodellierung des Software & Systems Engineering Lehrstuhls der Technischen Universität München, ist ein Werkzeug zur Unterstützung der automatischen Bewertung von Softwarequalität entstanden. Zur Früherkennung von Problemen während der Entwicklung sehen es Deißenböck und Wagner in (Deißenböck, 2008) als notwendig an, die kontinuierliche Verfolgung von Softwarequalität durch Werkzeuge zu unterstützen. Ergebnis ist das Werkzeug conQAT (Continuous Quality Assessment Toolkit). Initial entwickelt wurde das

Werkzeug zur Bewertung von Quellcode, wofür es existierende Code Analyser integriert und deren Ergebnisse weiter aggregiert. Neben dieser klassischen Code Analyse existieren auch Erweiterungen, um Matlab Simulink Modelle zu analysieren. In diesem Zusammenhang beschreiben die Entwickler der TU München (Deißenböck, 2008) auch die Notwendigkeit einer kontinuierlichen Qualitätsbewertung auf verschiedenen Granularitätsebenen und für verschiedene Design Artefakte.

Scheint eine Integration verschiedener existierender Code Analyser bei einem Fokus auf Quellcode noch sinnvoll zu sein, so ist dieses Vorgehen ungeeignet, wenn der Fokus der Qualitätsbewertung nicht nur auf ein Design Artefakt beschränkt ist. In der Elektronikentwicklung liefern verschiedene Methoden und Werkzeuge relevante Analyseergebnisse für die Qualitätsbewertung unterschiedlicher Design Artefakte, die für eine Prozessüberwachung jedoch alle berücksichtigt werden müssen. Eine Integration dieser Methoden und Werkzeuge ist nicht praktikabel, wenn nicht sogar unmöglich. So existieren z.B. bei der Entwicklung integrierter Schaltungen deutlich mehr Werkzeuge, die relevanten Output erzeugen. Ferner fehlt eine Anbindung an die Projekt- bzw. Prozesssicht, zeitlich bezogene Qualitätsanforderungen sind nicht modellierbar. Insgesamt adressiert das Werkzeug die **Ziele 5 und 6**, nicht aber die anderen Ziele dieser Arbeit.

2.3.4 Diskussion

Die in 2.3.2 vorgestellten Qualitätsmodelle weisen allein im Softwarebereich eine hohe Heterogenität auf. Sowohl die ISO/IEC Standards als auch darauf basierende Modelle bzw. deren Komponenten können keinen Anspruch auf Allgemeingültigkeit erheben (vgl. (Broy, 2005)). Dies bestätigt die in Kapitel 1 formulierte Hypothese der Nichtexistenz eines einheitlichen Qualitätsmodells, sowie die notwendige Anpassung von Qualitätsmodellen auf Unternehmens- bzw. Projektkontext.

Keines der vorgestellten Qualitätsmodelle kann mehr als ein definiertes Ziel dieser Arbeit erfüllen. Dies ist insofern nicht weiter verwunderlich, denn letztendlich stellen Qualitätsmodelle auch nur ein weiteres Werkzeug für den Nachweis von Qualitätseigenschaften dar und sind auf einer Ebene mit anderen typischen V&V Techniken zu nennen. Sie zeichnen sich jedoch besonders dadurch aus, dass sie sich mit oftmals vernachlässigten Anforderungen befassen, diese strukturieren und bewerten. Gemeinsam ist allen Qualitätsmodellen die Top-Down Strukturierung von abstrakten Kriterien hin zu messbaren Produktmetriken. Interessant ist zudem der Ansatz von Wagner und Deißenböck zur Integration verschiedener Qualitätsmodelle für eine ganzheitliche Bewertung. Jedoch fokussiert auch dieser Ansatz, wie alle Qualitätsmodelle, nicht funktionale Anforderungen und lassen funktionale Anforderungen außer Acht. Eine Erfassung zusammen mit funktionalen Anforderungen, bzw. eine Integration in den Requirements Engineering und angrenzende V&V Prozesse fehlt in der Regel. Nur unter Betrachtung aller Anforderungsarten wird eine wirkliche Qualitätsaussage und damit eine effektive Prozess- und Projektsteuerung ermöglicht.

Die in 2.3.3 diskutierten Qualitätsbewertungsframeworks beinhalten neben einem Ansatz zur Qualitätsmodellierung auch ein Vorgehensmodell zu deren entwicklungsbegleitenden Anwendung. Teilweise beinhalten sie auch Werkzeuge, um die Auswertungen zu automatisieren. Letztendlich leiden die Qualitätsbewertungsframeworks aber auch an den Unzulänglichkeiten existierender Qualitätsmodellierungsansätze. Die folgende Tabelle fasst die Zielerfüllung zusammen.

	Z1: Einheitliche Qualitätsdefinition	Z2: Effiziente Projekt-spezifische Auswertung	Z3: Integration V&V Ergebnisse	Z4: Domänenübergreifende Qualitätsbetrachtung	Z5: Entwicklungsbegleitende Auswertung	Z6: Berücksichtigung von Prozess- und Projektkontext	Z7: Automatische Datenerfassung und -integration	Z8: Anpassbarkeit an bestehende Entwicklungsprozesse
Klassische Ansätze								
Dromey				X				
ISO9126/ISO2 50xx Serie								
Aktivitätsbasierte Qualitätsmodelle	(X)	(X)						
ISO14598								
EMISQ					X		X	
EQM		(X)				(X)		
SQUID	(X)	(X)				(X)		
conQAT				(X)	X		X	

Tabelle 5: Zielerfüllung Ansätze der Qualitätsmodellierung

Eine Forschungsagenda (Wagner, 2010) von Qualitätsexperten aus Forschung und Praxis, fasst u.a. folgende Herausforderungen für Softwarequalität zusammen:

1. *Anpassungen von Qualitätsmodellen:* Software wird in unterschiedlichen Domänen angewandt, was Einfluss auf die Anforderungen an die zu entwickelnden Systeme hat. Qualitätsanforderungen müssen in Einklang mit zugehörigen Qualitätsmodellen

definiert werden. Modelle wie die ISO 9126 bieten keinerlei Richtlinien, wie sie auf unterschiedliche Domänen und Projekte anzupassen sind. was die Operationalisierung von Qualitätsmessungen schwierig macht.

2. *Allgemein vs. speziell*: Es besteht die Schwierigkeit eine optimale Balance zwischen Wiederverwendung und Spezialisierung von Qualitätsmodellen zu finden. Die Verwendung von Standards wie der ISO9126 minimieren die Aufwände, sind aber nicht für spezielle Anwendungskontexte optimiert. Auf der anderen Seite erfordern auf ein Projekt spezialisierte Modelle einen enormen Aufwand, sind aber am genauesten. Die Wahrheit liegt wie bei vielen Dingen in der Mitte, die es zu finden gilt.
3. *Prozessmodelintegration*: Die Forschungsagenda nennt die „Integration“ von Qualitätsmodellen in das verwendete Prozessmodell als eine Herausforderung. Es ist zu definieren, wann es zum Einsatz kommen soll und welche Ziele zu dem Zeitpunkt gesetzt sind. Auch wenn nicht vollständig klar ist, was die Autoren unter einer „Integration“ in das Prozessmodell verstehen, so scheint auch hier die Notwendigkeit gesehen zu werden, eine Verknüpfung zwischen beiden Sichten zu schaffen, um so ein effizientes Prozesscontrolling zu erreichen.
4. *Einführung von Qualitätsmodellen:* Die Einführung von Qualitätsmodellen im Rahmen von Qualitätsinitiativen in Unternehmen ist verschiedenen Schwierigkeiten ausgesetzt. Initiativen sind oft langfristig ausgelegt, so dass unter den Mitarbeitern auf Grund der anfänglichen Mehrarbeit der unmittelbare Mehrwert nur selten erkannt wird. Ebenfalls Einfluss auf die Motivation hat die bereits genannte Inflexibilität von Qualitätsmodellen für die Qualitätsbewertung. Durch die fehlende projektspezifische Anpassung leidet Effizienz und Effektivität.
5. *Werkzeugunterstützung für die Qualitätsmessung:* Im Zusammenhang mit der Messung von Softwarequalität mit Qualitätsmodellen fehlt es immer noch an geeigneten Messwerkzeugen, die vor allem eine leichte und effiziente Anpassung der Qualitätsmessung an den unternehmensinternen Kontext erlauben.

Die Herausforderungen der Forschungsagenda fokussieren zwar die Softwaredomäne, lassen sich aber auch auf andere Domänen übertragen, in denen die Forschungslandschaft hinsichtlich der Qualitätsmodellierung schwächer ausgeprägt ist.

2.4 Zusammenfassung und Handlungsbedarf

Nach Einführung in die Begrifflichkeiten und den Stand der Technik des Projekt- und Qualitätscontrollings sowie des quantitativen Messens in Abschnitt 2.1 untersuchte dieses Kapitel Ansätze des Requirements Engineerings mit Fokus auf der Anforderungsdefinition und -verfolgung. Bereits der gegebene Überblick lässt die vielfältigen Möglichkeiten zur Anforderungsdefinition erahnen. Auf ihre Art definieren alle einen dedizierten Teil der gewünschten Systemqualität. Es existiert de facto keinen alles abdeckenden Spezifikationsmechanismus. Vielmehr kombinieren Requirements Engineering Prozesse verschiedene Ansätze mit unterschiedlichen Formalisierungsgraden und Fokus (CESAR, 2010d), um eine möglichst breite Abdeckung des Problemraums zu erreichen. Bei

einer Qualitätsbewertung sind im Prinzip alle definierten Anforderungen zu berücksichtigen.

Neben Ansätzen zur Anforderungsdefinition wurden diverse akademische Arbeiten und kommerzielle Werkzeuge zur Verfolgung von Anforderungsverfeinerungen, Implementierung und Verifikation vorgestellt. Die existierenden modellbasierten Ansätze ermöglichen eine detaillierte projektspezifische Erfassung dessen, was als „Qualitätsziele“ für das System und deren Komponenten definiert wurde. Ferner erlauben sie eine entwicklungsbegleitende Verfolgung aller anfallenden Analyseergebnisse. Zusammenfassend: Metamodelle zur Anforderungsverfolgung sind ein Baustein innerhalb dieser Arbeit. Ob jedoch ein existierendes Metamodell eins zu eins übernommen werden kann oder aber Erweiterungen notwendig werden, wird im Konzeptkapitel im Detail erarbeitet.

Neben Lösungen zur Anforderungsdefinition und –verfolgung wurde der Stand der Technik im Bereich Qualitätssicherung speziell aus Sicht der Elektronikentwicklung beschrieben und die Vielzahl an Verifikationsmethoden sowohl für funktionale als auch nicht funktionale Anforderungen dargestellt. Besonders intensiv untersuchte das Kapitel Ansätze der Qualitätsmodellierung und –bewertung, von denen in den letzten 30 Jahren auf den ersten Blick viele verschiedene entstanden sind, die sich jedoch im Wesentlichen nur in ihrer konkreten Ausprägung der verwendeten Metriken unterscheiden. Der grundsätzliche Aufbau gleicht sich in den meisten Fällen. Abstrakte Charakteristika werden hin zu quantitativen Metriken verfeinert. Seit den 90er Jahren sind darüber hinaus verschiedene Qualitätsbewertungsframeworks entstanden, welche Qualitätsmodelle zur Qualitätsbewertung verwenden. Keines dieser existierenden Ansätze erfüllt die formulierten Ziele dieser Arbeit. Tatsächlich sind aktuelle Arbeiten zur Qualitätsmodellierung und –bewertung auf einer Ebene mit anderen Verifikationstechniken zu sehen. Es existiert aktuell kein Ansatz, der diese hochgradig verteilten Informationen systematisch zusammenträgt, sie für die Prozessteuerung aufbereitet und gegen eine einheitliche prozessorientierte Qualitätsdefinition abgleicht.

Zusammenfassend lässt sich aus dem Stand der Technik der folgende Handlungsbedarf ableiten, auf deren Grundlage in Kapitel 3 das Lösungskonzept entwickelt wird:

- *Prozessorientierte Bewertung:* Nahezu alle vorgestellten Ansätze zur Qualitätsbewertung sind produktorientiert. Selbst wenn alle Daten an einer Stelle verfügbar wären, besteht immer noch Bedarf zur Interpretation und Operationalisierung für die Prozesssteuerung. Aus Sicht des Projektcontrollings wiederum besteht Bedarf an einer regelmäßigen Bewertung des Fortschritts basierend auf aktuellen Entwicklungsdaten. Diese Lücke zwischen Produkt und Prozess gilt es zu schließen, um eine effektive prozessorientierte Bewertung zu ermöglichen und den im Prinzip vorhandenen aber verborgenen Informationsschatz zu heben. Nur so ist **Ziel 6** adressierbar.
- *Systematische Anpassung auf konkrete Projekte:* Measurement Standards und Metriken existieren viele, oft ist es jedoch die Herausforderung diese Standards auf das eigene Unternehmen und eben auch auf einzelne Projekte anzuwenden.

Diese Kalibrierung kann jedoch schnell zu aufwändig werden, wenn sie für jedes Projekt immer wieder durchgeführt werden muss. Es besteht Bedarf an einem systematischen Vorgehen, dass diese Problematik adressiert. Dies ist für eine entwicklungsbegleitende Qualitätsbewertung und zur Adressierung von **Ziel 2** dieser Arbeit unabdingbar.

- *Einheitliche Sicht auf qualitätsrelevante Daten:* Sowohl seitens definierter Anforderungen als auch seitens des Nachweis der Anforderungserfüllung herrscht eine hohe Heterogenität, die sich durch zahlreiche Medienbrüche wiederspiegelt. Es existieren sowohl verschiedene Möglichkeiten zur Beschreibung von Anforderungen und eine wahrscheinlich noch größere Menge an angewandten Verifikationsmethoden, die jede für sich nur einen Teilaspekt betrachtet. Aus technologischer Sicht, ist das Problem der Integration von Entwicklungsdaten zwar prinzipiell gelöst. Es besteht jedoch weiterhin Bedarf an einer einheitlichen Sicht auf Anforderungen für eine einheitliche Qualitätsdefinition (**Ziel 1**) und V&V Ergebnisse (**Ziel 3**), um eine Qualitätsbewertung gemäß Definition in Abschnitt 1.2.1 zu ermöglichen. Die bisher existierenden Metamodell basierten Ansätze sind entweder
  - *Nicht umfassend genug*, d.h. sie betrachten nur bestimmte Design Artefakte (z.B. Anforderungsdefinition und/oder –verfolgung),
  - *oder dienen primär einem anderen Anwendungszweck*. EAST-ADL2 dient eher dem Zweck der Architekturbeschreibung, denn als Integrationsansatz speziell als Grundlage für ein entwicklungsbegleitendes Qualitätscontrolling.
- *Domänenübergreifende Betrachtung:* Existierende Qualitätsbewertungsframeworks fokussieren sich im Wesentlichen auf die Bewertung von Software bzw. Software Quellcode. Es besteht Bedarf an methodischen Ansätzen, welche eine abstraktionsebene übergreifende Qualitätsüberüberwachung erlaubt, wie sie z.B. in der Elektronikentwicklung (siehe **Ziel 4**) notwendig ist.
- *Werkzeugunterstützung:* Nicht zuletzt besteht Bedarf an Werkzeugunterstützung, um den bisher genannten Handlungsbedarf und die **Ziele 5 und 7** für eine entwicklungsbegleitende und automatische Auswertung, sowie die **Ziel 8** zur Anpassung an beliebige Entwicklungsprozesse zu adressieren.

3 Prozessorientiertes Produktqualitätsmonitoring

Die bisherigen Kapitel legten Problemstellung und den Stand der Technik umfassend dar, um die Grundlage zur Konzeption einer Methodik für ein entwicklungsbegleitendes prozessorientiertes Produktqualitätsmonitoring für die Entwicklung elektronischer Systeme zu schaffen. Ausgehend vom identifizierten Handlungsbedarf, stellt dieses Kapitel die Spezifikation des Ansatzes vor. Zu diesem Zweck führt Abschnitt 3.1 zunächst die zentrale Idee eines prozessorientierten Produktqualitätsmonitorings ein und verfeinert entlang der Ausführung die Anforderungen an das Konzept. Abschnitt 3.2 erläutert wie das entwickelte Qualitätsmodellierungsframework die definierten Anforderungen umsetzt. Darauf aufbauend beschreibt Abschnitt 3.3 das Vorgehensmodell zur Anwendung des entwickelten Qualitätsmodellierungsframeworks. Der letzte Abschnitt liefert schließlich eine Zusammenfassung.

3.1 Idee

Allgemein gesprochen erfordert ein entwicklungsbegleitendes Qualitätsmonitoring, welches die Ergebnisse für die Projektsteuerung operationalisiert folgendes: die Erfassung des Solls in Form von Qualitätszielen und den Abgleich mit aktuellen Ist-Daten gemessen durch definierte Qualitätsmetriken. Die beiden Begriffe sind dabei wie folgt definiert:

Definition (Qualitätsmetrik): *Eine Qualitätsmetrik misst den Ist Wert einer definierten Produktqualitätseigenschaft zu einem Zeitpunkt in der Entwicklung. Eine Qualitätsmetrik macht dabei keine Einschränkung über die Granularität der Qualitätseigenschaft. Sie kann sowohl direkt messbar als auch über Aggregationsregeln berechnet werden.*

Definition (Qualitätsziel): *Ein Qualitätsziel definiert einen Zielwert für eine Qualitätsmetrik, welches zu einem festgelegten Zeitpunkt erreicht sein muss. Qualitätsziele dienen als Fertigstellungskriterien für Tasks und Meilensteine.*

Hierbei berücksichtigt das Soll zwar notwendigerweise definierte Anforderungen, allein ist dies jedoch nicht hinreichend. Um eine systematische Planung, Überwachung und Optimierung von Entwicklungsprojekten zu ermöglichen, ist bei der Definition von Qualitätsmetriken und -zielen eine prozessorientierte Sicht einzunehmen (siehe Abschnitt 2.4, **Handlungsbedarf „Prozessorientierte Qualitätsbewertung“**). Dies sei an folgenden Beispielen näher erläutert:

- Die Gesamtqualität setzt sich aus einer Vielzahl an Qualitätscharakteristika zusammen. Einzelne Phasen und Prozessschritte betrachten aber jeweils nur Teilaspekte. Anforderungsdefinition und -analyse betrachten Anforderungsqualitätsziele. Die Architekturentwicklung und –bewertung sucht ein Optimum zwischen konkurrierenden funktionalen und nicht-funktionalen Qualitätscharakteristika. In Implementierungsphasen stellen sich wieder andere Ziele.
- Anforderungen werden entlang des Entwicklungsprozesses mehrfach verifiziert, sowohl durch dieselbe, als auch durch unterschiedlichen Verifikationsmethoden. Sie unterscheiden sich lediglich in ihrem Verifikationsgegenstand, der unterschiedlichen Repräsentation des Produkts, die mit fortlaufender Entwicklungsdauer stetig konkreter wird. Sind die Ergebnisse in frühen Phasen noch mit einem höheren Unsicherheitsgrad behaftet, verringert sich dieser in späteren Phasen. In frühen Phasen ist die Forderung nach Übererfüllung oder Definition von Toleranzen bei noch unsicheren Bewertungen zur Unterstützung der Risikominimierung denkbar. Die Anforderung bleibt über die Zeit dieselbe, der Prozesskontext fügt eine neue Dimension hinzu.
- Qualitätsziele sind aus Projektsicht immer zeitpunktbezogen. Je nach Projektablauf definieren z.B. Zwischenmeilensteine, dass ein bestimmter Verifikationsgrad bereits drei Monate vor dem Endmeilenstein erreicht sein muss, um diesen einzuhalten.

Abbildung 21 fasst die beschriebene Grundidee noch einmal zusammen: Qualität wird ausgehend von Tasks definiert. Ein Task repräsentiert eine atomare Aufgabe, die ein- oder mehrfach in einer oder mehreren Prozessen anwendbar ist. Über die Granularität einer Task wird dabei keine Einschränkung gemacht. Eine Qualitätsmetrik misst eine Qualitätscharakteristik bzw. ein Qualitätsattribut wie z.B. den Anforderungserfüllungsgrad. Eine prozessorientierte Qualitätsdefinition berücksichtigt Informationen wie Deadlines, Toleranzen und Übererfüllung und sogar die Verifikationsmethode zum Nachweis der Anforderungserfüllung. Ergebnis ist eine prozessorientierte Interpretation von Produktdaten, dargestellt durch die Lupe. Ein Prozess besteht aus mehreren Tasks, die alle durch die „Prozesslupe“ auf das Produktmodell bzw. einen Teil davon blicken und dieses entsprechend definierter Qualitätsmetriken interpretieren. Die Ergebnisse sind durch ihre Annotation an die Tasks direkt für die Projektsteuerung nutzbar.

Eine Aussage über die Gesamtqualität bzw. über den Projektfortschritt lässt sich aus den Ergebnissen der Qualitätsmetriken und -ziele aller Tasks eines Prozesses ableiten. Zur Prozessteuerung, aber auch zur Effizienzbewertung einzelner Methoden, sind Aussagen auf Task Ebene notwendig, so dass sich die Arbeit auf die Berechnung dieser Werte konzentriert. Die Berechnung einer verdichteten Kennzahl über den Gesamtentwicklungsstand ist an dieser Stelle jedoch ohne weiteres möglich.

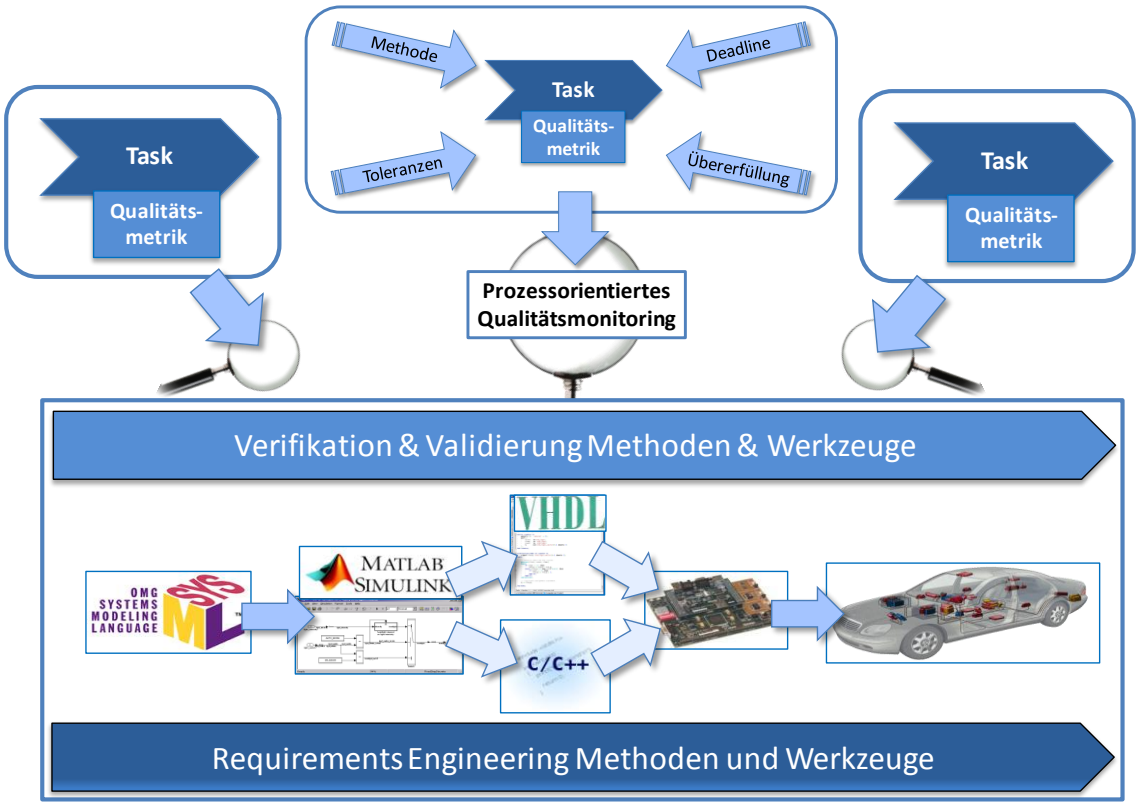

Abbildung 21: Prozessorientiertes Qualitätsmonitoring - die Idee

Die Notwendigkeit der Prozessorientierung zeigt sich auch in Beispielen aus der Praxis. So beschreibt Scheible (Scheible, 2011) ein Qualitätsmodell, welches Teilergebnisse aus verschiedenen Prozessschritten zusammenführt und aggregiert. Das resultierende Qualitätsmodell ist jedoch statisch. Es ist unflexibel gegenüber Prozessänderungen und stellt kein systematisches, auf andere Unternehmen übertragbares Vorgehen dar. Eine prozessorientierte Qualitätsdefinition behebt diese Probleme. Folgende Anforderungen an die Modellierung der Qualitätsziele (Soll-Modell) existieren:

- ***Anforderung 1*** *– Konzepte zur Prozessdefinition.* Die Abbildung eines Entwicklungsprozesses inkl. Konzepte wie Tasks und Meilensteine ist zu unterstützen.
- ***Anforderung 2*** *– Konzepte zur Definition von Qualitätsmetriken und –zielen.* Es sind Konzepte notwendig, die eine prozessbezogene Definition von Qualitätsmetriken und –zielen ausgehend von einzelnen Tasks erlauben.
- ***Anforderung 3*** *– Konzept zur (losen) Integration von Prozess- und Produktsicht.* Das Konzept muss die definierten Prozesse und Qualitätsmetriken mit der Produktsicht in Beziehung setzen, um Qualitätsmetriken basierend auf definierten Anforderungen und V&V Ergebnissen möglichst automatisch auszuwerten.

Eine prozessorientierte Qualitätsdefinition ist Ausgangspunkt für eine effiziente Kalibrierung definierter Qualitätsziele auf einzelne Projekte (siehe Abschnitt 2.4, **Handlungsbedarf „Systematische Anpassung auf einzelne Projekte“**). Folgende Anforderungen stellen sich an das Konzept hinsichtlich der Kalibrierung einer Qualitätsdefinition auf ein Projekt:

- ***Anforderung 4 –*** *Prozessorientierte Kalibrierung der Qualitätsmessung auf einzelne Projekte.* Der Entwicklungsprozess bestimmt maßgeblich die Qualitätsdefinition und –bewertung. Die Methodik muss sicherstellen, dass bei einer Anpassung des Entwicklungsprozesses auch die Qualitätsdefinition angepasst wird.
- ***Anforderung 5 –*** *Gewährleistung einer stringenten Anforderungsorientierung für ein zielorientiertes Messen.* Anforderungen sind die Basis für Qualitätsbewertungen einzelner Systemkomponenten, Architekturen oder ganzer Systeme. Sie definieren die gewünschte Qualität aus Produktsicht (siehe Definition in 1.2.1). Jede prozessorientierte Qualitätsbewertung fußt auf definierten Anforderungen und deren Allokation an Systemkomponenten. Dies gewährleistet eine projektspezifische Auswertung.

Sowohl der letzte Punkt, als auch die Produktmodell Darstellung in Abbildung 21 deuten die zentrale Rolle von Anforderungen an. Auch eine prozessorientierte Qualitätsbewertung fußt auf definierten Anforderungen. Neben dem bereits motivierten Soll-Modell wird daher eine geeignete Repräsentation dieser und weiterer qualitätsrelevanten Produktdaten benötigt (siehe Abschnitt 2.4, **Handlungsbedarf „Einheitliche Sicht auf qualitätsrelevante Daten“**), um definierte Qualitätsziele mit dem aktuellen Ist-Stand abzugleichen. Nachfolgende Anforderungen stellen sich an die Repräsentation der Produktdaten (Ist-Modell):

- ***Anforderung 6 –*** *Konzepte zur Repräsentation von Anforderungen.* Es ist eine einheitliche Sicht über alle Anforderungsarten, ob funktional oder nicht funktional, ob informal oder formal zu erstellen. Alle Anforderungen sind zentraler Bestandteil einer Qualitätsbewertung. Existierende Ansätze ermöglichen dies aktuell nicht.
- ***Anforderung 7 –*** *Konzepte für die Anforderungsverfolgung.* Es sind Konzepte zur Anforderungsverfolgung hinsichtlich ihrer Verfeinerung und Formalisierung, ihrer Implementierung und ihrer Validierung und Verifikation notwendig.
- ***Anforderung 8 –*** *Konzepte zur Abbildung von Analysen und Verifikationsergebnissen.* Es ist eine einheitliche Sicht auf heutzutage über mehrere Quellen verteilte V&V Ergebnisse zu schaffen und eine Analyse des aktuellen Qualitätsstands zu ermöglichen.

Hierbei gilt es, nicht nur eine Repräsentation des Produkts in die Qualitätsbewertung mit einzubeziehen, wie es vielfach in der Softwarequalitätsbewertung der Fall ist. Jede für den Entwicklungsprozess relevante Repräsentation des Produkts ist nach Möglichkeit zu unterstützen (siehe Abschnitt 2.4, **Handlungsbedarf „Domänenübergreifende Betrachtung“**). Daraus lassen sich weitere Anforderungen ableiten:

- ***Anforderung 9 –*** *Berücksichtigung verschiedener Partialmodelle.* Das Qualitätsmodellierungsframework muss die Abbildung beliebiger Produktpartialmodelle ermöglichen, die im betrachteten Entwicklungsprozess erstellt werden.
- ***Anforderung 10 –*** *Erweiterungsmechanismen für verschiedene Partialmodelle*. Da die betrachteten Produktpartialmodelle zwischen Domäne, Unternehmen und auch

Projekten variieren, muss das Qualitätsmodellierungsframework entsprechende Erweiterungs- bzw. Konfigurationsmechanismen zur Verfügung stellen.

Um schließlich eine effiziente automatisierte Qualitätsbewertung zu gewährleisten, gilt es die folgenden Anforderungen bezüglich notwendiger Werkzeugunterstützung zu berücksichtigen (siehe Abschnitt 2.4, **Handlungsbedarf „Werkzeugunterstützung“***)*:

- ***Anforderung 11 –*** *Formale Modellgrundlage.* Grundlage des Qualitätsmodellierungsframeworks sind formale Metamodelle, um basierend auf den Konzepten eine entsprechende Werkzeugunterstützung zu implementieren.
- ***Anforderung 12 –*** *Automatische entwicklungsbegleitende Datenerfassung, -integration und –analyse.* Für eine vollständige Werkzeugunterstützung sind alle Schritte von der Datenerfassung, über die Datenintegration bis hin zur Ausführung der Qualitätsbewertung und der Darstellung ihrer Ergebnisse werkzeugseitig umzusetzen.
- ***Anforderung 13 –*** *Anbindung an unternehmensinterne IT.* Das Qualitätsmodellierungsframework und die auf dessen Basis definierten Bewertungen sind mit Entwicklungsdaten zu füllen, die in meist proprietären Datenquellen und –formaten vorliegen. Da dies regelmäßig durchzuführen ist und Entwickler nicht bei ihrer täglichen Arbeit behindert werden dürfen, wird ein Konzept zur Anbindung an existierende Entwicklungswerkzeuge und –datenbanken benötigt. Da dieses Thema nicht Schwerpunkt dieser Arbeit ist und hierfür schon verschiedene Ansätze existieren, werden an entsprechenden Stellen in diesem Konzeptkapitel die verschiedenen Ansätze diskutiert und eine geeignete Lösung für die prototypische Implementierung ausgewählt.

Der weitere Verlauf dieses Kapitels beschreibt den entwickelten Lösungsansatz, welcher entsprechend der Beitragsformulierung im ersten Kapitel aus zwei Bestandteilen besteht:

1. Ein Qualitätsmodellierungsframework für ein prozessorientiertes Qualitätsmonitoring, welches sowohl die prozessorientierte Qualitätsdefinition (Soll-Modell) als auch die Erfassung aller qualitätsrelevanten Produktdaten (Ist-Modell) zusammenführt. Die so geschaffene einheitliche Sicht auf über mehrere Werkzeuge verteilte Produkt- und Prozessdaten ermöglicht eine entwicklungsbegleitende Qualitätsauswertung. Darüber hinaus unterstützt es die Kalibrierung auf verschiedene Unternehmenskontexte, was ihm den Charakter eines Frameworks verschafft.
2. Ein Vorgehensmodell für ein entwicklungsbegleitendes prozessorientiertes Produktqualitätsmonitoring. Es beschreibt a) die notwendigen Schritte zur Anpassung des entwickelten Qualitätsmodellierungsframeworks auf Anwendungsdomäne und Unternehmen, b) die notwendigen Schritte zur Anbindung der existierenden IT Infrastruktur und eingesetzten Entwicklungswerkzeuge und c) die notwendigen Schritte zur Kalibrierung unternehmensweiter Qualitätsbewertungsdefinitionen auf einzelne Projekte.

3.2 Prozessorientiertes Qualitätsmodellierungsframework

Dieser Abschnitt beschreibt den Kern des prozessorientierten Qualitätsmonitorings. Die Arbeit verwirklicht die einleitend skizzierte Idee mit Hilfe einer einheitlichen Modellierungsbasis. Das im weiteren Verlauf detaillierte Modellierungsframework stellt diese Basis dar und führt eine prozessorientierte Qualitätsdefinition mit ursprünglich verteilt vorliegenden Produktdaten zusammen. Es beinhaltet domänen- und unternehmensübergreifend gültige Konzepte eines prozessorientierten Qualitätsmonitorings, sowie Möglichkeiten der Kalibrierung auf einen unternehmensspezifischen Entwicklungsprozess. Abbildung 22 stellt eine Übersicht des Modellierungsframeworks dar. Modellkonzepte sind dabei in hellblau, zugehörige Auswertungen in dunkelblau dargestellt. Es besteht aus den folgenden Bestandteilen:

Produktmetamodell: Das Produktmetamodell (Abschnitt 3.2.1) erfasst den Ist-Stand eines Projekts und stellt die Grundlage einer prozessorientierten Qualitätsbewertung dar. Es schafft die benötigte einheitliche Sicht auf qualitätsrelevante Daten. Dies umfasst Konzepte zur Repräsentation von Anforderungen (**Anforderung 6**), deren Verfeinerung, Verknüpfung zum Design und ihrer Validierung und Verifikation (**Anforderung 7**), sowie V&V Ergebnissen (**Anforderung 8**). Einmal integriert, ermöglichen diese Daten eine projektspezifische Qualitätsbewertung basierend auf Anforderungsqualität und -erfüllung (**Anforderung 5**). Für die notwendige Integration von Daten aus Computer-Aided-Engineering- und Analysewerkzeugen wird auf die gängige Praxis existierender Ansätze zurückgegriffen, ein integriertes Produktmodell durch die lose Kopplung der Elemente ihrer Metamodelle zu schaffen. Beispiele dieses Ansatzes sind die Arbeit von Hausmann (Hausmann. 2009) oder EAST-ADL2, eine Beschreibungssprache für Elektronikarchitekturen im Auto (ATESST, 2010).

Auch wenn das grundlegende Vorgehen ähnlich ist, so haben die genannten Ansätze unterschiedliche Anwendungszwecke. EAST-ADL2 ist eine Architekturbeschreibungssprache aus dem Automotive Bereich und damit mehr eine (domänenspezifische) Designsprache, als explizit auf die Qualitätsbewertung zugeschnitten. Hausmann stellt ein generelles Konzept zur Integration von Produktdaten vor, um beliebige Metriken zu definieren und auszuwerten. Er macht keinerlei Einschränkung hinsichtlich der enthaltenen Konzepte und Semantik der Modelle und deren Verknüpfung. Eine Definition von Kernkonzepten ist jedoch notwendig, um Qualitätsbewertungen auf eine einheitliche semantische Basis zu stellen.

Die logische Konsequenz ist daher die Verfeinerung des Ansatzes nach Hausmann durch die Einführung eines Core Produktmetamodells als Grundlage eines prozessorientierten Qualitätsmonitorings.

Prozessmetamodell: Neben dem Produktmetamodell beinhaltet das Modellierungsframework ein Prozessmetamodell (Abschnitt 3.2.2). Das Prozessmetamodell erlaubt die Abbildung existierender Prozessmodelle (z.B. Rational Unified Process, V-Modell XT) und unternehmensinterne Entwicklungsprozesse, deren einzelne Tasks, deren Abfolge

und Arbeitsergebnisse (**Anforderung 1**). Ferner ermöglicht es eine prozessorientierte Definition von Qualitätsmetriken (**Anforderung 2**), welche die aktuelle Qualität und damit den Fortschritt einzelner Tasks Prozesskontext abhängig bewertet. Darüber hinaus unterstützt das Qualitätsmodellierungsframework die Integration zwischen dem Prozess- und dem Produktmetamodell (**Anforderung 3**, Abschnitt 3.2.3), um a) die Qualitätsmetriken im Prozessmetamodell basierend auf der formalen Beschreibung des Produktmetamodells durchzuführen und b) während der Entwicklung eindeutig zu erkennen, welche aktuell vorhandenen Design Artefakte innerhalb des Produktmetamodells gegen die prozessorientierte Qualitätsdefinition auszuwerten sind. Ferner legt es die Grundlage für eine prozessorientierte Kalibrierung der Qualitätsdefinition (**Anforderung 4**).

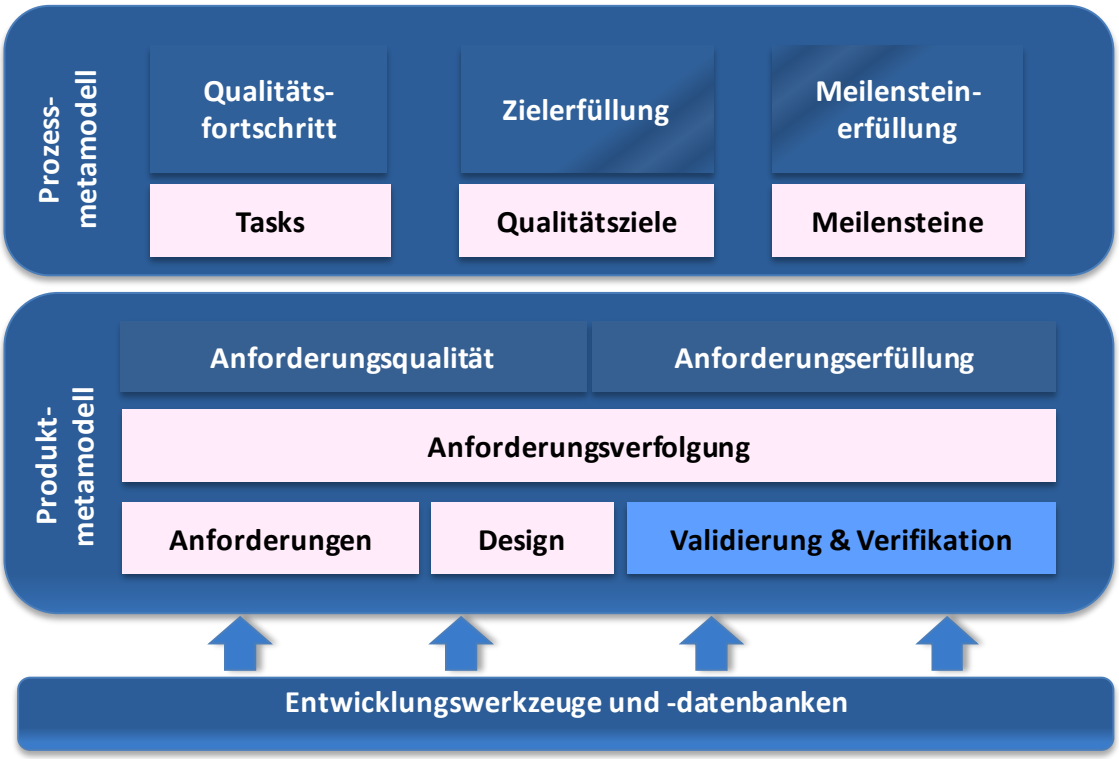

Abbildung 22: Übersicht Qualitätsmodellierungsframework

Das Framework bietet ebenfalls die Möglichkeit bestehende Entwicklungswerkzeuge und Datenquellen anzubinden, um das Modell mit aktuellen Entwicklungsergebnissen zu füllen. Modelltransformationen stellen die Verbindung zwischen dem Framework und den Entwicklungswerkzeugen her. Unabhängig von der konkreten Umsetzung folgt die Arbeit hier dem Ansatz, Abbildungen zwischen den proprietären Formaten der Ausgangsdaten und dem im nächsten Abschnitt beschriebenen Produktmetamodell zu entwickeln. Zur Umsetzung der Abbildungen lassen sich existierende Technologien anwenden. So entwickelte Hausmann (Hausmann, 2009) in seiner Arbeit ein Konzept zur Überführung verteilter Produktdaten in eine einheitliche Repräsentation mittels Modelltransformationen. Das Fraunhofer Institut FOKUS entwickelte eine Bus Technologie mit dem Namen ModelBus (Hein, 2009), (Armengaud, 2011), welche es erlaubt unter

Zuhilfenahme von Adaptern und Metamodellen heterogene Daten und Modelle auszutauschen. Dies sind nur zwei denkbare Umsetzungen. Für eine detaillierte Beschreibung sei auf die genannten Quellen verwiesen.

Die drei nächsten Abschnitte beschreiben den Aufbau des Qualitätsmodellierungsframeworks.

3.2.1 Produktmetamodell

Dieser Abschnitt definiert Produktmetamodellkonzepte, um Anforderungsqualität und –erfüllung entwicklungsbegleitend zu bestimmen und die Grundlage für ein prozessorientiertes Qualitätsmonitoring zu schaffen. Zu diesem Zweck untersucht der Abschnitt zunächst, wie sich Anforderungsqualität und –erfüllung berechnen, um daraus die notwendigen Konzepte für das Produktmetamodell abzuleiten. Für beide Aufgaben existiert eine Vielzahl an Techniken (siehe Abschnitt 2.3), deren Anwendungsmöglichkeit nicht zuletzt auch vom Grad der Formalisierung, Art und Fokus der Anforderung abhängt. Diese Arbeit ersetzt diese Techniken nicht, sondern nutzt und interpretiert deren Ergebnisse. Diese Prämisse prägt die folgende Beschreibung.

Berechnung Anforderungsqualität

Sowohl einzelne als auch eine Anforderungsmenge besitzen Qualitätsattribute (siehe 2.3.1). Viele der Attribute basieren auf Anforderungsverfolgungskonzepten (z.B.: „Sind alle Anforderungen an eine Komponente verknüpft?“, „Sind alle Anforderungen der oberen Ebene berücksichtigt?“). Andere Attribute lassen sich wiederum mittels Reviews oder formalen Analysemethoden absichern (z.B.: „Sind meine Anforderungen widerspruchsfrei?“, „Sind meine Anforderungen eindeutig?“). Die Gesamtanforderungsqualität lässt sich dann über ein gewichtetes arithmetisches Mittel über alle Attribute berechnen.

Berechnung Anforderungserfüllung

Die Berechnung bildet V&V Ergebnisse auf einen Wert zwischen 0 und 1 ab, den Erfüllungsgrad einer Anforderung. Dies ist aus zwei Gründen herausfordernd. Zum einen existieren verschiedene Anforderungsarten und zum anderen diverse Verifikationsmethoden für den Nachweis, dass die entwickelte Lösung die Anforderung korrekt und ohne Fehler umsetzt. Die Berechnung kann sich daher je nach Anforderung unterscheiden. Um die Berechnung zu systematisieren, führt dieses Kapitel zunächst eine für diese Arbeit gültige Kategorisierung von Anforderungen ein, da die Art einer Anforderung maßgeblich die Messung des Erfüllungsgrads beeinflusst. Das dies schwieriger ist als es klingt, zeigt eine Untersuchung von Glinz (siehe (Glinz, 2005), (Glinz, 2007)). Seine Veröffentlichungen sind Grundlage der folgenden Betrachtung.

In der Literatur existieren verschiedene Klassifikationssysteme von Anforderungen. Typisch ist die Unterscheidung zwischen funktionalen und nicht-funktionalen Anforderungen (siehe z.B. (IEEE, 1998), (Kotonya, 1998), (Lamsweerde, 2001). Davies (Davis, 1992) macht eine ähnliche Unterscheidung, verwendet jedoch die Begriffe

“Behavioural” vs. “non-behavioural“. Nach Sommerville (Sommerville, 2004) beschreiben Funktionale Anforderungen „was das System machen soll“, alle anderen sind nicht-funktional. Was ist jedoch eine nicht-funktionale Anforderung? Die Antwort auf diese Frage unterscheidet sich jedoch je nach Quelle. Davis (Davis, 1992) bezeichnet diese als Qualitäten und verwendet Boehms Qualitätsdefinition (Boehm, 1976). Der IEEE Standard 830-1993 über Softwareanforderungsspezifikationen (IEEE, 1998) unterscheidet nicht-funktionale Anforderungen in Anforderungen an externe Interfaces, Performance Anforderungen, Attribute und Design Constraints, wobei Attribute Qualitäten wie Zuverlässigkeit, Verfügbarkeit, Sicherheit, etc. beschreiben. Der IEEE Standard Glossary of Software Engineering Terminology (IEEE, 1990) erwähnt den Begriff nicht-funktionale Anforderung gar nicht. Hinzu kommt, dass solche Kategorisierungen eine eindeutige Einordnung von Anforderungen oft nicht erlauben. So klassifizieren mehrere Kategorisierungen z.B. Security Anforderungen als nicht funktional. Letztendlich sind Security Mechanismen aber auch nichts anderes als Funktionen. Man könnte also auch von funktionalen Anforderungen sprechen. Ähnliches gilt z.B. auch für Safety Anforderungen.

Nach Glinz (Glinz, 2005) vermischen existierende Kategorisierungen Aspekte wie Art oder Repräsentation von Anforderungen und tragen so zu dem geschilderten Problem bei. Er schlägt daher ein mehrdimensionales Kategorisierungsschema vor:

- **Kind**: Klassifiziert Anforderung nach deren Art (z.B. funktional, Performance, etc.)
- **Representation**: Darstellung der Anforderung. Diese bestimmt maßgeblich, wie eine Anforderung verifizierbar ist. Operationale Anforderungen beschreiben funktionales Verhalten und sind z.B. mittels Tests oder Review bewertbar. Quantitative Anforderungen werden gemessen, qualitative sind nicht direkt verifizierbar und sind weiter zu verfeinern und deklarative Anforderungen sind nur durch Review eines Designers verifizierbar.
- **Satisfaction**: Diese Klassifizierung unterscheidet Anforderungen nach Art der Erfüllung. Hier gibt es harte oder weiche Grenzen. Im Gegensatz zur harten Anforderungen, können weiche Anforderungen auch teilweise erfüllt werden.
- **Role**: Unterscheidet Anforderungen in „klassische“ Anforderungen (prescriptive), was das System machen soll und Anforderungen, die vom System nicht kontrollierbar (assumptive) sind.

In (Glinz, 2007) stellt Glinz eine mögliche Anforderungstaxonomie vor (siehe Abbildung 23), die sich auf die Dimension „Kind“ bezieht.

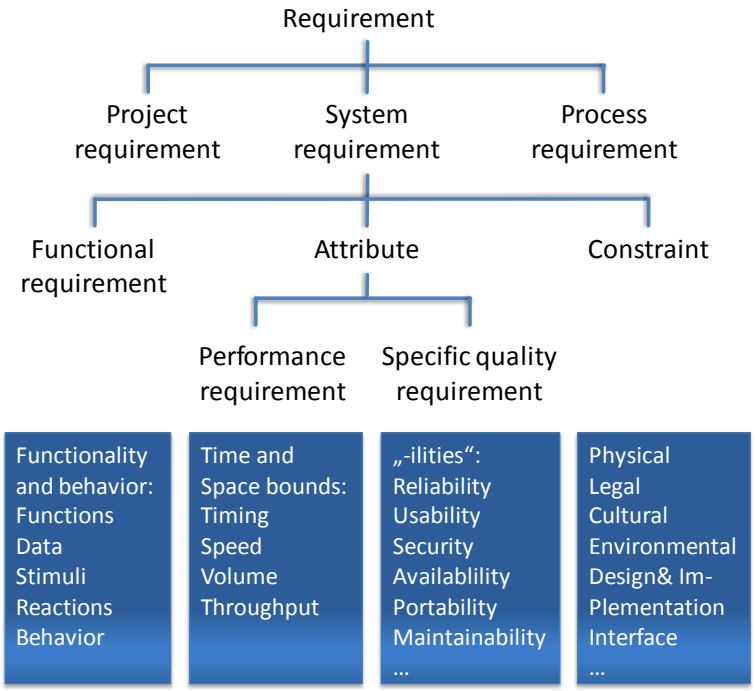

Abbildung 23: Anforderungstaxonomie nach (Glinz, 2007)

Diese Arbeit folgt der Argumentation von Glinz und wählt einen ähnlichen Ansatz zur Anforderungskategorisierung. Neben den beiden Kategorien „Representation“ und „Kind“, wird die Anforderungserfüllungsberechnung durch den Gültigkeitsbereich der Anforderung bestimmt. Die Art der Komponente beeinflusst die Anforderungserfüllungsberechnung, wenn sich die Verifikation derselben Eigenschaft bei unterschiedlichen Komponentenarten unterscheidet.

Demzufolge existieren für die Berechnung des Anforderungserfüllungsgrads die folgenden Kontextparameter:

- **Art der Anforderung:** Unterschiedliche Anforderungsarten (funktional, Wartbarkeit, Design Constraints wie Gewicht oder Kosten) erfordern unterschiedliche Verifikationsmethoden und damit potentiell eine unterschiedliche Interpretation, ob und inwieweit die Anforderung erfüllt ist. Es existiert keine Einschränkung bezüglich der verwendeten Taxonomie. Hier ist sowohl die Kategorie von Glinz, oder der von Standards wie der ISO 25000 oder der ISO/IEC WD 29148.3 verwendbar.
- **Repräsentation der Anforderung:** Gültige Repräsentationen sind operational, quantitativ und qualitativ.
- **Gültigkeitsbereich der Anforderung**: Der Gültigkeitsbereich einer Anforderung bestimmt sich durch die Systemkomponente (Hardware-, Software-, funktionale Architekturkomponente), welches sie umsetzen soll.

Entsprechend der Belegung dieser drei Kontextparameter lässt sich eine passende Erfüllungsgradfunktion auswählen. Ausgangspunkt ist dabei die Dimension „Repräsentation“:

Operational - Da operationale Anforderungen in der Regel funktional sind und sich durch Reviews oder Tests verifizieren lassen, berechnet sich der Erfüllungsgrad aus dem Verifikationsstatus allokierter Testfälle (siehe Abbildung 24) und optional aus Testabdeckungsinformationen (wenn vorhanden), welche Informationen über die Testqualität selbst liefern.

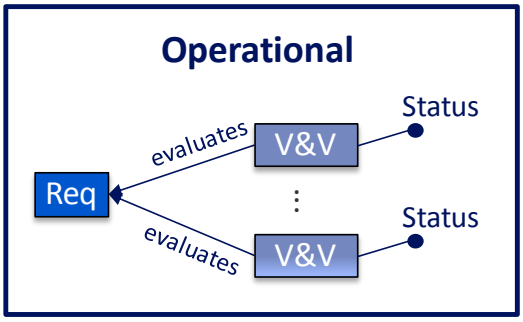

Abbildung 24: Konzepte zur Berechnung der Anforderungserfüllung bei operationalen Anforderungen

Quantitativ - Quantitative Anforderungen werden durch Messungen verifiziert. Beispiele hierfür sind Metriken und Simulationen, die z.B. das Zeitverhalten von Komponenten bestimmen. Auch Kosten, Gewicht, Energieverbrauch oder Fläche (typische Design Constraints) sind quantitative Anforderungen und werden durch verschiedene Abschätzungsverfahren bestimmt. Vereinfacht besteht auch hier die Möglichkeit ausschließlich auf den Status des Tests zurückzugreifen, was aber lediglich einen Wert von 0 oder 1 liefert. Für eine genauere Berechnung ist bei quantitativen Anforderungen der aktuelle Wert mit Zielwert und optionalen Toleranzwerten ins Verhältnis zu setzen (siehe Abbildung 25).

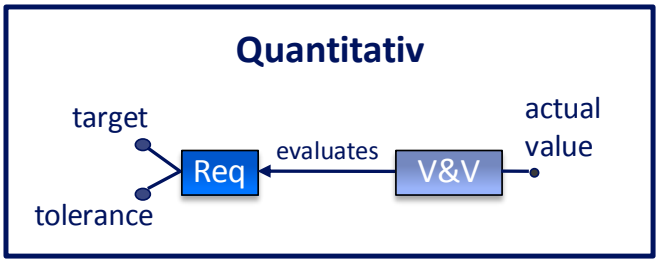

Abbildung 25: Notwendige Konzepte zur Berechnung der Anforderungserfüllung bei quantitativen Anforderungen

Qualitativ – Da qualitative Anforderungen nicht direkt verifizierbar sind, werden an dieser Stelle Qualitätsmodelle zur Beschreibung der jeweiligen Qualitätscharakteristika verwendet. Um diese effizient zu nutzen, sind sie bereits im Anforderungsdefinitionsprozess zu verwenden (Dörr, 2005). Qualitätsmodelle werden direkt durch die Anforderungsdefinition kalibriert und sind gleichzeitig Vorlage für eine vollständige Verfeine-

rung einer nicht-funktionalen Anforderung. Das Qualitätsmodell wird ebenfalls zur Bewertung der Anforderungserfüllung herangezogen. Auf unterster Ebene stehen direkt verifizierbare quantitative Anforderungen. Entlang der Verfeinerungsstruktur im Qualitätsmodell sind auch die abstrakten Qualitätscharakteristika (kurz QC) durch hinterlegte Aggregationsregeln bewertbar. Die Auswahl der korrekten Aggregationsregel erfolgt entsprechend der anderen beiden Kontextparameter Art und Gültigkeitsbereich der Anforderung.

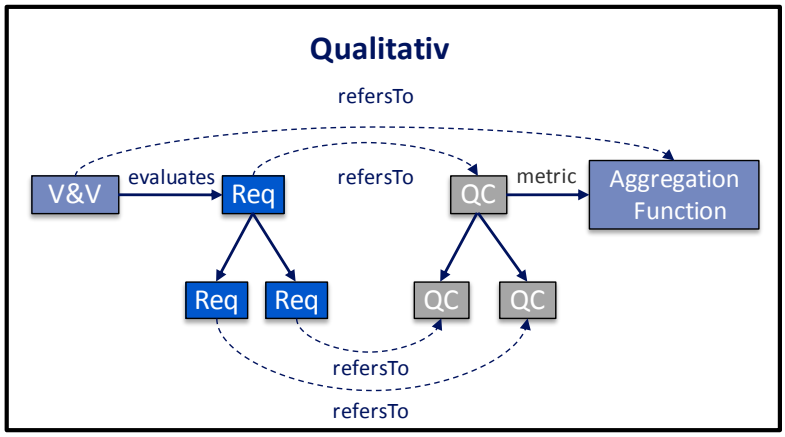

Abbildung 26: Konzepte zur Berechnung der Anforderungserfüllung bei qualitativen Anforderungen

Tatsächlich ist das Qualitätsmodell selbst die Verifikationsmethode einer qualitativen Anforderung. Die Berechnung ist an dieser Stelle eine Stufe detaillierter als bei den vorherigen Anforderungsrepräsentationen, bei denen ein Verifikationsergebnis bereits vorliegt.

Konzepte des Produktmetamodells

Das Produktmetamodell definiert die einzuhaltende Semantik aller nach Domäne und Unternehmen unterschiedlich spezialisierten Ausprägungen des Qualitätsmodellierungsframeworks. Entsprechend der eingeführten Berechnung für Anforderungsqualität und –erfüllung lassen sich verschiedene notwendige Konzepte ableiten. Dies umfasst Anforderungs-, Design- und V&V Konzepte (siehe Abbildung 27), deren Relevanz für die Qualitätsbewertung jeweils in der Konzeptdefinition erläutert wird.

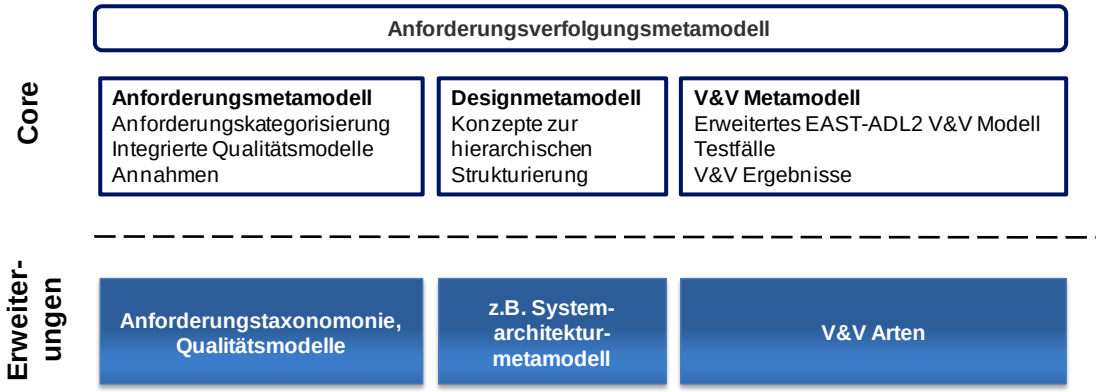

Abbildung 27: Übersicht Produktmetamodell für das Qualitätsmonitoring

Weiterhin bietet das Qualitätsmodellierungsframework die Möglichkeit, Metamodellerweiterungen (zum Beispiel Beschreibung von Systemarchitekturen wie in Abbildung 27 dargestellt) einzuführen und adressiert damit die **Anforderungen 9 und 10** aus der Einleitung. Je nach Rolle entwickelt das Unternehmen z.B. ganze Systeme, nur die Hardware oder Software einer einzelnen Komponente. Entsprechend wird über andere Prozessergebnisse gesprochen. Schlussendlich spielen die Konzepte des Produktmetamodells ebenfalls eine zentrale Rolle zur Integration des Produktmetamodells mit dem Prozessmetamodell (siehe Abschnitt 3.2.3).

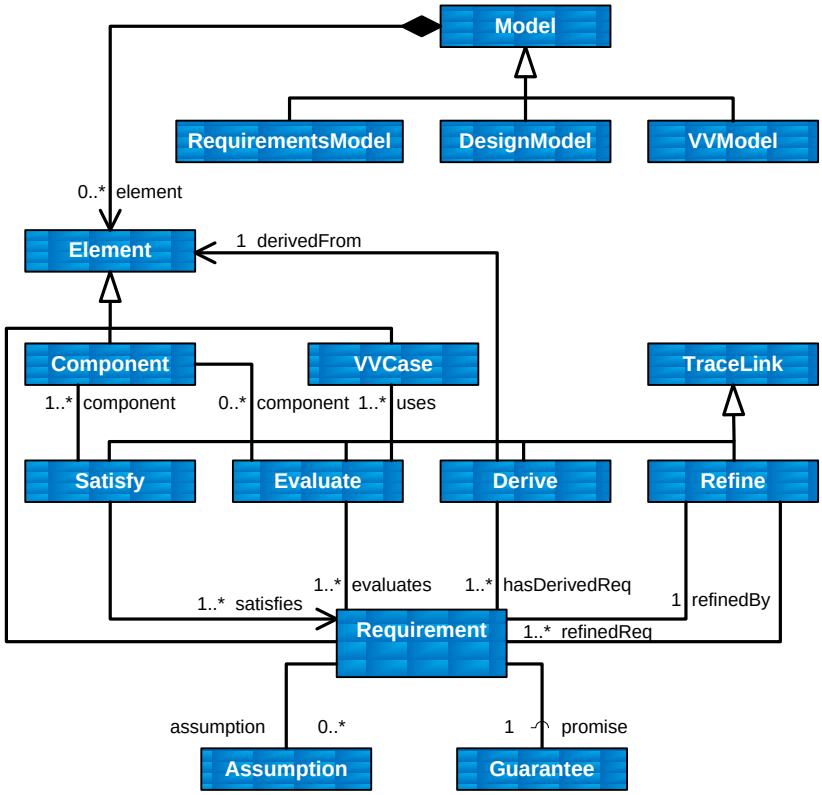

Abbildung 28: Basiskonzepte des Core Produktmetamodells

Die nächsten Absätze beschreiben die wesentlichen Konzepte des Core-Produktmetamodells. Das Core-Produktmetamodell basiert auf einer Genese existierender Metamodelle zur Anforderungsverfolgung. Entsprechende Verwandtschaften sind ebenfalls in den Konzeptbeschreibungen referenziert.

Abbildung 28 beschreibt alle Anforderungsverfolgungskonzepte, sowie die miteinander verknüpften Design Artefakte. Die folgende Tabelle beschreibt zunächst die allgemeinen Design Artefakte und deren (qualitätsrelevanten) Eigenschaften:

Konzept	Beschreibung
Model	Abstraktes Konzept zur Beschreibung von Modellen. Modelle enthalten Elemente.
Element	Abstraktes Konzept zur Beschreibung eines Elements. Alle einzel-

	nen Modellelemente müssen entweder direkt oder indirekt von diesem Element erben.
Requirements Model	Ein Anforderungsmodell ist eine Menge an Anforderungen und stellt das Ergebnis eines Anforderungsdefinitionsprozesses dar.
Requirement	Eine einzelne Anforderung eines Anforderungsmodells. Es existieren unterschiedliche Arten von Anforderungen beschrieben durch eine Anforderungstaxonomie. Die Anforderungstaxonomie ist an die jeweilige Domäne anpassbar.
Assumption	Die Annahme, auf der die Anforderung basiert. Entsprechend des Metamodells des EU Projekt SPEEDS[7], siehe z.B. (Damm, 2009b).
Guarantee	Beschreibt den eigentlichen Inhalt der Anforderung. Eine Eigenschaft, die durch das System garantiert wird, sollte die Anforderung erfüllt sein. Entsprechend des Metamodells des EU Projekt SPEEDS, siehe z.B. (Damm, 2009b).
DesignModel	Ein Design Modell beschreibt das Ergebnis von Design Schritten. Konkrete Ausprägungen sind z.B. eine Architekturbeschreibung oder eine Implementierung einer Hardware- oder Softwarekomponente.
Component	Eine Komponente ist ein Teil eines Designmodells. Komponenten sind hierarchisch angeordnet und definieren die Struktur des Designs für die jeweilige betrachtete Abstraktionsebene. Das Konzept der Komponente ist in domänenspezifischen Modellen zu konkretisieren (z.B. funktionale Architekturkomponente, Hardwarekomponente, Softwarekomponente, etc.) und stellt den Gültigkeitsbereich einer Anforderung dar.
VVModel	Ein V&V Modell beschreibt eine Menge an Testfällen. Dies ist unabhängig von der konkreten Testmethodik.
VVCase	Ein V&V Fall repräsentiert einen einzelnen Testfall. Es repräsentiert sowohl Analysen zum Nachweis eines Anforderungsqualitätsattributs als auch zum Nachweis der Anforderungserfüllung.

Tabelle 6: Konzepte des Core-Produktmetamodells

[7] SPEEDS – Speculative and Exploratory Design in Systems Engineering – www.speeds.eu.com

Die nächste Tabelle beschreibt die Konzepte der Anforderungsverfolgung, begründet deren Notwendigkeit für die Qualitätsbewertung und definiert Quelle und Ziel der jeweiligen Relation.

Konzept	Beschreibung
Refine	Verfeinerungen fügen Details zu einer bereits existierenden Anforderung (Quelle) hinzu ohne dabei deren Semantik zu verändern. Ergebnis ist eine neue Anforderung (Ziel), die über diese Relation mit der bereits existierenden Anforderung verknüpft wird. Eine typische Anwendung hierfür sind Anforderungsformalisierungen. Ebenfalls Bestandteil von SysML (SysML, 2009) und EAST-ADL2 (ATESST, 2010).
Derive	Verknüpft Anforderungen (Ziel) mit den Elementen (Quelle), aus der sie abgeleitet sind. Dies können Anforderungen der oberen Ebene in der Systemhierarchie, Design Entscheidungen oder durchgeführte Analysen sein. Über das (Nicht-) Vorhandensein dieser Relation lassen sich eine Reihe an Anforderungsqualitätsattributen bestimmen, welche auf Anforderungsverfolgungskonzepten beruhen. Ebenfalls Bestandteil von SysML (SysML, 2009) und EAST-ADL2 (ATESST, 2010).
Satisfy	Verknüpft Anforderungen (Ziel) mit Komponenten (Quelle) und sagt aus, dass eine Komponente die Anforderung erfüllen soll. Nachzuweisen ist dies durch Verifikationsmaßnahmen. Durch diese Relation wird gleichzeitig der Gültigkeitsbereich der Anforderung definiert, ein Auswahlparameter für die zu verwendende Anforderungserfüllungsfunktion. Siehe auch (SysML, 2009), (ATESST, 2010), (Ramesh, 2001), usw.
Evaluate	Verknüpft V&V Fälle (Quelle) mit evaluierten Anforderungen (Ziel). Dies beinhaltet sowohl den Nachweis geforderter Design Eigenschaften als auch die Analyse von Anforderungsqualitätsattributen. Die Relation findet sich ebenfalls in anderen Metamodellen (SysML, 2009), (ATESST, 2010), (Ramesh, 2001), heißt jedoch in der Regel „Verify“. Um Missverständnissen vorzubeugen, dass hiermit nur Verknüpfung zu Verifikationsmaßnahmen (aber nicht Validierungsmaßnahmen) repräsentiert werden, verwendet die Arbeit den Begriff `Evaluate`. Die Ergebnisse von mit Anforderungen verknüpften V&V Fällen sind Eingang für die Berechnung von Anforderungsqualität und –erfüllung.

Tabelle 7: Anforderungsverfolgungskonzepte des Core-Produktmetamodells

Aufbauend auf den Konzepten zur Anforderungsverfolgung und zur Beschreibung von Design Artefakten, beschreiben die nächsten Absätze das Anforderungs- und das V&V Metamodell genauer. Des Weiteren enthält das Core-Produktmetamodell ein minimales Designmetamodell, welches jedoch lediglich die hierarchische Dekomposition des Systems und seiner Komponenten definiert. Auf Grund des geringen Umfangs wird dieses nicht gesondert aufgelistet.

Anforderungsmetamodell

Das Anforderungsmetamodell enthält entsprechend den Anforderungen an die Berechnung des Anforderungserfüllungsgrads Konzepte zur Abbildung beliebiger Anforderungen und Anforderungsarten. Die konkreten Anforderungsarten sind je nach Domäne und Unternehmen während der Kalibrierung des Qualitätsmodellierungsframeworks hinzuzufügen. Die grundlegende Unterscheidung zwischen funktionalen Anforderungen, Qualitätsanforderungen und Constraints, wie sie sich auch bei Glinz (Glinz, 2007) wiederfindet, ist jedoch auch hier fest. Weitere Konzepte verknüpfen Qualitätsmodelle mit Anforderungen, um die Erfüllung von qualitativen Anforderungen zu bestimmen. Dies entspricht im Wesentlichen dem Ansatz der NFR Methode (Dörr, 2005). Abbildung 29 stellt Konzepte und Relationen des Metamodells dar. Es enthält alle Informationen, um eindeutig die Anforderungserfüllungsfunktion für jede Anforderungen zu bestimmen (siehe Beginn dieses Kapitels). Bisher existierende Anforderungsmetamodelle waren hierzu nicht in der Lage. Eine erste Grundlage für eine ganzheitliche Qualitätsbewertung ist gelegt.

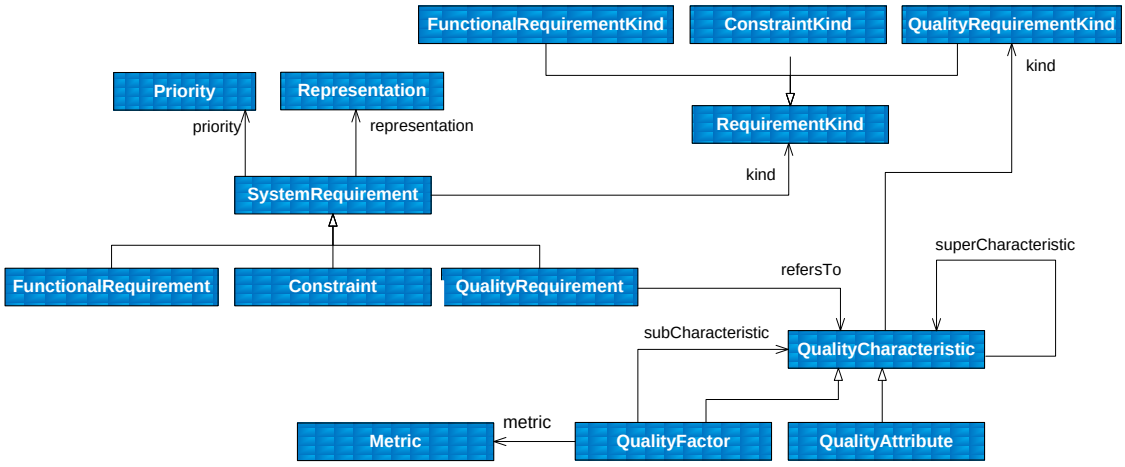

Abbildung 29: Konzeptionelle Darstellung des Anforderungsmetamodells

Tabelle 8 beschreibt die einzelnen Konzepte, die neben dem bereits beschriebenen SystemRequirement Bestandteil des Anforderungsmetamodells sind:

Konzept	Beschreibung
Functional Requirement	Eine Anforderung, welche durchzuführende Funktionen oder Aufgaben eines Systems oder Teilsystems beschreibt. Es stellt eine von drei festen Anforderungskategorien im Anforderungsmetamodell

	dar.
Constraint	Eine Anforderung, welche die Optionen für den Entwickler bei der Lösungsfindung durch die Einführung unwiderruflicher Beschränkungen verringert. Beispiel sind die Verwendung von bereits existierenden Komponenten, eine bestimmte gesetzte technologische Lösung oder physikalische Beschränkungen.
Quality Requirement	Eine Qualitätsanforderung definiert eine gewünschte Eigenschaft unter der das System operieren muss.
Priority	Die Priorität einer Anforderung.
Representation	Die Repräsentation der Anforderungen gemäß der gegebenen Repräsentation Dimension nach Glinz.
Requirement Kind	Die Art der Anforderung. Wird domänenspezifisch ausgestaltet.
Quality Characteristic	Eine abstrakte Klasse für alle Qualitätscharakteristika eines Qualitätsmodells. Wird von `QualityRequirement` referenziert.
QualityFactor	Ein spezielles Qualitätscharakteristikum in einem Qualitätsmodell, welches eine Qualitätscharakteristik weiter verfeinert. Der aktuelle Wert eines `QualityFactors` berechnet sich mit Hilfe einer definierten Metrik.
QualityAttribute	Eine spezielle Qualitätscharakteristik in einem Qualitätsmodell, welches nicht weiter verfeinerbar ist und ein direkt messbares Qualitätsattribut darstellt.
Metric	Eine Metrik berechnet den aktuellen Wert für einen `QualityFactor` basierend auf den Eingangswerten der Qualitätscharakteristika, auf denen der `QualityFactor` basiert.

Tabelle 8: Konzepte des Anforderungsmetamodells

V&V Metamodell

Das V&V Metamodell hält alle Validierungs- und Verifikationstätigkeiten und deren Ergebnisse nach und stellt damit das Gegenstück zu geforderten Eigenschaften (den Anforderungen) dar. Das dieser Arbeit zu Grunde liegende V&V Metamodell basiert auf dem V&V Metamodell aus EAST-ADL2, unterscheidet jedoch im Gegensatz zu EAST-ADL2 explizit zwischen Validierungs- und Verifikationsaktivitäten und fügt Konzepte zur Typisierung der Testarten ein, welche die Art der Analyse eindeutig bestimmen.

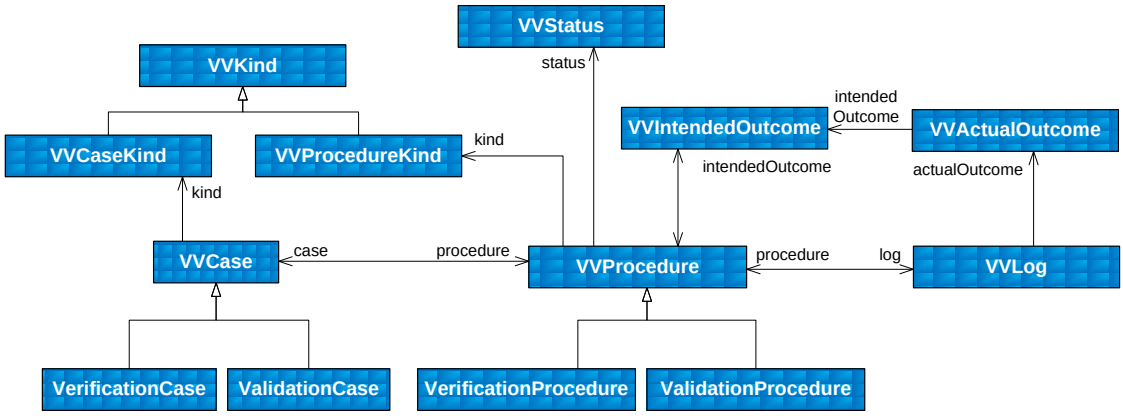

Abbildung 30: Konzeptionelle Darstellung des V&V Metamodell

Tabelle 9 beschreibt die einzelnen Konzepte, die neben dem bereits beschriebenen `VVCase` Bestandteil des V&V Metamodells sind:

Konzept	Beschreibung
Verification Case	`VerificationCase` repräsentiert eine Verifikationsaktivität zur Entwurf, Implementierung und Produkt Verifikation. Ziel ist der Nachweis, dass die jeweiligen Design Artefakte ihre Anforderungen korrekt umsetzen.
Validation Case	Ein `ValidationCase` repräsentiert eine Analyse, ob das richtige Produkt gebaut wird. Hierzu gehört der Nachweis, dass die Anforderungen valide (d.h. konsistent, vollständig, komplett, usw.) sind, sowie der Nachweis, dass das fertige Produkt sich wie erwartet verhält.
VVProcedure	Eine `VVProcedure` repräsentiert einen einzelnen Analyseschritt in einer Validierungs- oder Verifikationsaktivität, um das Ziel der jeweiligen Gesamtaktivität zu erreichen. Jede `VVProcedure` definiert das erwartete Ergebnis nach Ausführung dieses einzelnen Analyseschritts.
Verification Procedure	Eine `VerificationProcedure` repräsentiert einen Analyseschritt in einem `VerificationCase`. Die `VerificationProcedure` ist für eine konkrete Testumgebung und bestimmte Stimuli definiert. Hierfür gespeicherte Ergebnisse sind zentraler Input für die Berechnung von Anforderungserfüllungsgraden und Design Qualität.
Validation Procedure	Eine `ValidationProcedure` repräsentiert einen einzelnen Analyseschritt in einem `ValidationCase` zum Nachweis der Anforderungsqualität.

VVLog	Ein `VVLog` repräsentiert die Ergebnisse der Ausführung einer `VVProcedure`. Wenn z.B. ein Test am Freitag durchgeführt wird, und dann zur Ergebnisüberprüfung erneut am Montag, besitzen alle `VVProcedure` Objekte zwei `VVLog` Instanzen, welche die Testergebnisse von Freitag und Montag beschreiben.
VVIntended Outcome	Ein `VVIntendedOutcome` beschreibt die zu erwartenden Zielwerte der Ausführung einer `VVProcedure`.
VVActual Outcome	`VVActualOutcome` beschreibt das tatsächliche Ergebnis einer `VVProcedure`. Es sollte üblicherweise die erwarteten Zielwerte (spezifiziert durch `VVIntendedOutcome`) erfüllen, und ist demnach zur Bestimmung des `VVStatus` mit diesem abzugleichen.
VVStatus	Beschreibt den Status einer `VVProcedure`. Mögliche Werte sind z.B. „Erfolgreich", „Fehlgeschlagen", „Fehlerhaft" oder „Ausstehend".
VVKind	Ein `VVKind` wird zu Typisierung von `VVCases` und `VVProcedures` verwendet. Da die verwendeten V&V Methoden prozessspezifisch sind, redet man an dieser Stelle nur von diesem abstrakten Konzept. Es ist Teil der Kalibrierung des Qualitätsmodellierungsframeworks gültige `VVKind` Definitionen hinzuzufügen. Ein `VVKind` beschreibt sowohl Analysen zur Anforderungsvalidierung, wie Konsistenz oder Vollständigkeitsüberprüfungen, als auch Methoden wie Sicherheitsanalysen oder anforderungsbasiertes Testen.
VVCase Kind	Spezialisierung von `VVKind`. Beschreibt `VVCase` Typen.
VVProcedure Kind	Spezialisierung von `VVKind`. Beschreibt `VVProcedure` Typen.

Tabelle 9: Konzepte des V&V Metamodells

Zusammen bilden die vorgestellten Konzepte das integrierte Produktmetamodell. Bei der Entwicklung des Produktmetamodells verwendete die Arbeit zu einem hohen Anteil existierende Lösung und Standards und führt diese zusammen. Punktuell wurden neue Konzepte hinzugefügt, um Schwachstellen der existierenden Lösungen zu beheben. Die Grundlage der in dieser Arbeit entwickelten Qualitätsbewertung ist gelegt.

3.2.2 Prozessmetamodell

Wie bereits in der Einleitung des Konzeptkapitels erläutert, ist zur Realisierung einer prozessorientierten Qualitätsdefinition zweierlei erforderlich: Konzepte zur Beschreibung von Entwicklungsprozessen (siehe **Anforderung 1**) und eine darauf aufbauende prozessbezogene Definition von Qualitätsmetriken und –zielen (siehe **Anforderung 2**).

Im Laufe der Jahre sind eine Reihe unterschiedlicher Ansätze zur Prozessmodellierung entstanden. So sind z.B. die Structured Analysis and Design Technique (SADT) von Ross (Ross, 1977) und ihr Ableger IDEF (Integrated Definition) von Mayer (Mayer, 1995), zwei Modelle zur Softwaremodellierung, die auch zur Prozessmodellierung verwendbar sind. Auf Grund ihrer vielfältigen Einsatzmöglichkeiten sind jedoch die enthaltenen Konzepte sehr generisch. Ebenfalls weit verbreitet sind Ereignisgesteuerte Prozessketten (EPK), die 1992 von Scheer (Scheer, 1992) entwickelt wurden und zur Definition von Geschäftsprozessen verwendbar sind. Da bereits verschiedene Prozessmodellierungssprachen existieren, war es Bestreben dieser Arbeit, soweit wie möglich auf existierende Arbeiten und Standards aufzubauen und lediglich punktuell zu erweitern. Darüber hinaus war die Existenz eines expliziten Metamodells Voraussetzung, um eine möglichst einfache Verknüpfung mit dem Produktmetamodell zu ermöglichen. Die genannten klassischen Ansätze eigneten sich daher nicht. Die Metamodell Anforderung führte jedoch schnell zu zwei Standards der Object Management Group (kurz OMG):

- *Business Process Modelling Notation* (kurz BPMN): BPMN ist eine graphische Notation und beschreibt Workflows innerhalb eines Geschäftsprozesses. Entwickelt wurde der Standard durch die Business Process Management Initiative und wird durch die OMG fortgeführt. Der Standard liegt aktuell in der Version 2.0 (OMG, 2011) vor. Das Hauptziel von BPMN ist eine standardisierte, gut lesbare Notation zur Beschreibung von Workflows für alle Beteiligten, seien es Geschäftsprozess Designer oder diejenigen, die sie implementieren. Finales Ziel ist die Übersetzung in eine ausführbare Workflow Sprache wie BPEL (OASIS, 2007) oder XPDL (WMC, 2008).
- *Software and Systems Process Engineering Meta-Model* (kurz SPEM): SPEM ermöglicht es Organisationen ein konzeptionelles Framework zur Beschreibung von Entwicklungsmethoden und –prozessen zu definieren. Der Standard liegt mittlerweile in Version 2.0 (OMG, 2008) vor. Die eigentliche Ausführung von Prozessen ist nicht im Fokus von SPEM.

Sowohl SPEM als auch BPMN ähneln sich. Sie beschreiben Prozessabläufe mittels der Kombination einzelner Tasks. Im Gegensatz zu SPEM fokussiert BPMN auf die Ausführung von Workflows und adressiert somit in ihren Beschreibungsmöglichkeiten eine deutlich technisch orientiertere Ebene der Prozesse. Insgesamt ist das BPMN Vokabular vergleichbar mit UML Aktivitätsdiagrammen (Aldazabal, 2008). Im Gegensatz zu BPMN berücksichtigt SPEM z.B. die Beschreibung von Rollen und Work Breakdown Strukturen (Aldazabal, 2008). Vor allem aber erlaubt SPEM im Gegensatz zu BPMN die Definition einer Ontologie, bzw. einer Methoden Bibliothek, deren Elemente die

Grundlage jedweder Prozessdefinition bilden. Diese explizite Unterscheidung zwischen Methoden und Prozessen manifestiert sich in folgenden zwei Paketen:

- *Method Content*: Das Methoden Paket wird zur Beschreibung von Prozesselementen wie Arbeitsschritte (`TaskDefinition`), deren In- und Output (`WorkProductDefinition`), Rollen (`RoleDefinition`) und Entwicklungswerkzeuge (`ToolDefinition`) verwendet. Die Elemente beschreiben Methoden unabhängig von konkreten Prozessen, in denen sie eingesetzt und mehrfach verwendbar sind. Das Hauptkonzept ist `TaskDefinition`. Es repräsentiert den bisher als Task bezeichneten atomaren Prozessbaustein.
- *Processes with Methods*: Das Paket wird zur Beschreibung von konkreten Prozessen verwendet und verknüpft die durch das vorherige Paket beschriebenen Methodenbausteine. Prozesse werden durch die Verwendung von `TaskDefinition` Instanzen definiert.

Weder SPEM noch BPMN erlauben eine formale Definition von Qualitätsmetriken und -zielen, geschweige denn deren Auswertung basierend auf konkreten Entwicklungsdaten. Gegen die Verwendung von BPMN spricht jedoch die zu starke Workflow Orientierung. Die Definition von Qualitätsmetriken eignet sich eher auf globaler Ebene, denn auf der detaillierten technischen Workflow Ebene, in denen einzelne Schritte bis runter zu „Speichere Daten im SVN Repository“ reichen können. Eine Ebene auf der die Definition von Qualitätsmetriken und –zielen üblicherweise nicht angelangt. Auf Grund dessen und der Unterteilung zwischen Methoden- und Prozessdefinition, die den Faktor Wiederverwendung von Prozessbausteinen unterstützt, wird in dieser Arbeit SPEM als zu Grunde liegendes Prozessmetamodell ausgewählt.

Qualitätsmetriken und -ziele

Die Unterteilung in Methoden und Prozessdefinition eignet sich ebenfalls für die Definition von Qualitätsmetriken und -zielen. Eine Erweiterung des *Method Content* Pakets ermöglicht die Definition solcher Konzepte unabhängig vom konkreten Prozess- und Projektkontext und deren (Wieder-) Verwendung in verschiedenen Prozessen. Die Arbeit erweitert SPEM um generische Messkonzepte, wie sie Garcia (Garcia, 2009) beschreibt (siehe Abbildung 31). Die Definition von Qualitätsmetriken, repräsentiert durch das Konzept `MeasureDefinition`, basiert auf dem Produktmetamodell und den dort beschriebenen Konzepten und Eigenschaften. Die Definitionen sind so zu parametrisieren, dass sie gegen jedes konkrete Design Artefakt bzw. Modell der definierten In- und Outputs (siehe `WorkProductDefinition`) auswertbar sind.

Die Definition von Qualitätsmetriken reicht von einem `BaseMeasure`, der einen direkt messbaren Parameter aus dem Modell ausliest, bis hin zu komplexen Indikatoren. Qualitätsmetriken werden als hierarchische Baumstrukturen definiert, wie es auch in anderen Bereichen, wie z.B. bei der Qualitätsmodellierung (siehe Abschnitt 2.3.2) oder im Bereich des Software Measurements (siehe z.B. (Garcia, 2009)) der Fall ist. Die Hierarchien beschreiben die Abhängigkeiten zwischen den verschiedenen `MeasureDe-`

finition Arten, sowie die Aggregation von den direkt messbaren BaseMeasure Objekten hin zu DerivedMeasure und Indicator Objekten.

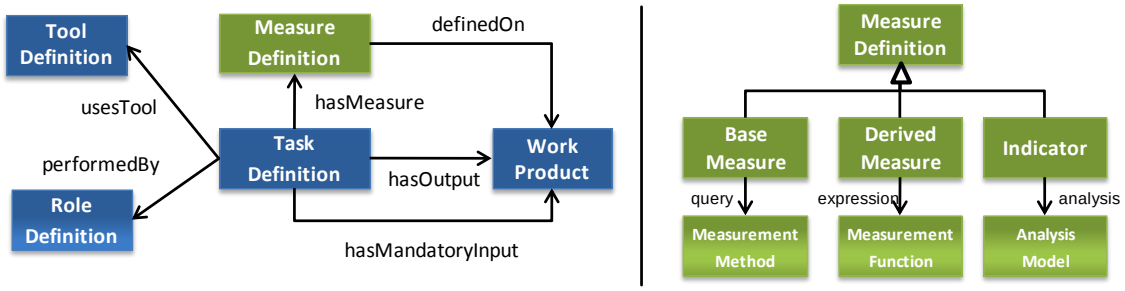

Abbildung 31: (a) Erweiterung existierender Konzepte mit MeasureDefinition. (b) Measurement Konzepte und deren Abhängigkeiten

Tabelle 10 beschreibt die Erweiterung im Detail. Die Definition der einzelnen Konzepte ist eng angelehnt an (Garcia, 2009). Übernahme von Definitionen sind gekennzeichnet:

Konzept	Beschreibung
Measure Definition	MeasureDefinition ist eine abstrakte Klasse für alle weiteren speziellen Measure Konzepte (z.B. BaseMeasure). Eine MeasureDefinition wird von einem oder mehreren Tasks referenziert und repräsentiert eine Qualitätsmetrik.
BaseMeasure	Ein BaseMeasure repräsentiert eine direkt messbare Qualitätsmetrik. Die Messung wird mittels Abfrage (MeasurementMethod) definiert, siehe (Garcia, 2009).
Derived Measure	Ein DerivedMeasure repräsentiert eine Qualitätsmetrik, welche aus anderen Qualitätsmetriken, repräsentiert durch Base- und DerivedMeasures, abgeleitet ist. Gemessen wird ein DerivedMeasure mit einer mathematischen Funktion (MeasurementFunction), siehe (Garcia, 2009).
Indicator	Ein Indicator repräsentiert eine Qualitätsmetrik, die ebenfalls aus anderen Qualitätsmetriken abgeleitet ist, jedoch im Gegensatz zum DerivedMeasure komplexere Analysemodelle (z.B. Bayes Klassifikation) für die Messung verwendet, siehe (Garcia, 2009).
Measurement Method	Eine Abfrage auf einem gegebenen Modell zur Messung eines bestimmten Attributs.
Measurement Function	Eine mathematische Funktion, um die Ergebnisse von zwei oder mehreren MeasureDefinition Objekten zu einem neuen Ergebnis zu kombinieren, siehe (Garcia, 2009).

Analysis Model	Ein Modell zur Bewertung eines `Indicator` Objekts, z.B. eine Bayes Klassifikation.

Tabelle 10: Erweiterung SPEM – Method Content Paket

Analog zum *Method Content* Paket erweitert die Arbeit ebenfalls das Paket *Process with Methods*, um definierte Qualitätsmetriken in verschiedenen Prozessdefinitionen wiederzuverwenden. Ein im Prozess verwendeter Task (`TaskUse`) besitzt Referenzen auf definierte Qualitätsmetriken, die der jeweiligen `TaskDefinition` der Methodenbibliothek zugeordnet sind. Auf Basis dieser Qualitätsmetriken sind Qualitätsziele (`Goal`) definierbar. Sie werden entweder mit einem Meilenstein (`Milestone`) verknüpft, an dem das Ziel erfüllt sein muss, oder sie definieren die Bedingungen nach denen ein bestimmter Task in einem Prozess erfolgreich durchgeführt wurde. Ein Qualitätsziel definiert einen Zielwert für eine Qualitätsmetrik sowie optionale minimale und maximale Toleranzen oder geforderte Übererfüllung. Die Kombination von Qualitätsmetriken und –zielen gibt Projektmanagern eine Vielzahl an Planungsmöglichkeiten und erlaubt die Überwachung des Qualitätsfortschritts über die Zeit. Eine vereinfachte Darstellung der Konzepte und Relationen findet sich in der folgenden Abbildung:

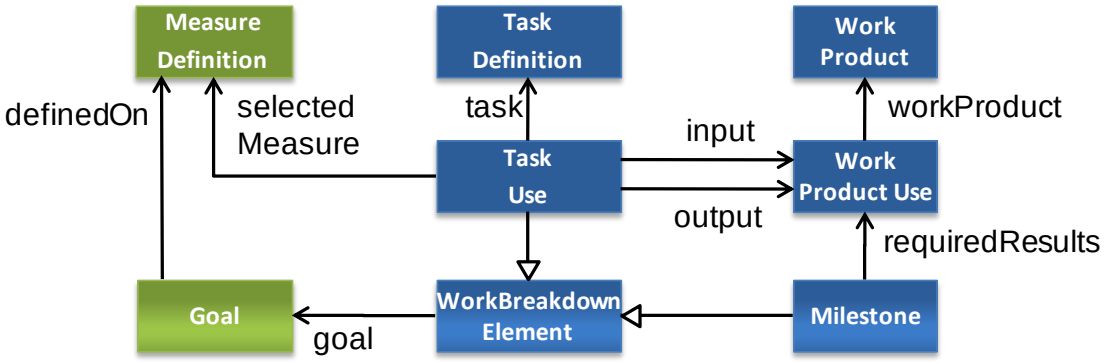

Abbildung 32: SPEM Erweiterung zur Beschreibung von Qualitätszielen

Tabelle 11 beschreibt Konzepte, Relationen und Eigenschaften der Erweiterung im Detail. Handelt es sich um ein in SPEM bereits vorhandenes Konzept, werden nur die Erweiterungen beschrieben:

Konzept	Beschreibung
TaskUse	*Siehe SPEM 2.0 Spezifikation.* Erweitert um eine Relation „selectedMeasure“ zur Verknüpfung von `MeasureDefinition` Instanzen. Standardmäßig deckt sich das mit den `MeasureDefinition` Instanzen, die auch schon mit der zugehörigen `TaskDefinition` verknüpft sind. Es besteht jedoch die Möglichkeit, nur eine Untermenge an Qualitätsmetriken der `TaskDefinition` zu verwenden.

Work Breakdown Element	*Siehe SPEM 2.0 Spezifikation.* Erweitert um eine Relation zur Definition von Qualitätszielen (`Goal`). Sowohl `TaskUse` als auch `Milestone` sind Spezialisierungen von `WorkBreakdownElement`. Dementsprechend sind sowohl Task als auch Meilenstein spezifische Qualitätsziele definierbar.
Goal	Ein Qualitätsziel definiert einen Zielwert für eine Qualitätsmetrik (`MeasureDefinition`) und optionale minimale und maximale Toleranzen oder geforderte Übererfüllung. Die Definition von Zielen ist ein weiteres Mittel, eine Qualitätsdefinition für bestimmte Prozesse oder Projekte zu verfeinern.

Tabelle 11: Erweiterung SPEM – Processes with Methods Paket

Die SPEM Erweiterung unterstützt sowohl die unternehmensweite Definition und Wiederverwendung von Qualitätsmetriken und -zielen sowie Quality Gate bzw. Meilensteinplänen, als auch deren spezielle Anwendung und Erweiterung für ein Projekt. Der Ansatz ist dabei nicht an eine spezielle Entwicklungsphase oder -ergebnis gebunden, sondern ist von verschiedenen Beteiligten der Wertschöpfungskette anwendbar, seien es Systemintegratoren oder Zulieferer. Vorausgesetzt wird lediglich, dass das Produktmetamodell alle notwendigen Konzepte zur Beschreibung des jeweiligen zu entwickelnden Produkts und seiner Zwischenergebnisse besitzt.

Um die prozessorientierte Qualitätsdefinition und -bewertung basierend auf dem Produktmetamodell und den dadurch repräsentierten Entwicklungsdaten durchzuführen, gilt es eine Integration dieser beiden Sichten herbeizuführen. Einen entsprechenden Lösungsansatz beschreibt der folgende Abschnitt.

3.2.3 Integration

Der vorherige Abschnitt schafft die Möglichkeit, Qualitätsmetriken wie Anforderungserfüllung und -qualität aus Prozesssicht zu definieren. Um nun die Definition und Auswertung auf Basis konkreter Entwicklungsdaten durchzuführen, ist eine Verknüpfung zwischen Produkt- und Prozessmetamodell zu schaffen. Das Ziel ist dabei kein monolithisches Modell, sondern vielmehr eine lose flexible Kopplung dieser beiden Sichten (siehe **Anforderung 3**). Weiterhin ermöglicht die Integration auch eine prozessorientierte Kalibrierung der Qualitätsmessung auf einzelne Projekte (siehe **Anforderung 4**). Dieser Abschnitt beschreibt die in dieser Arbeit entwickelten Konzepte zur Integration entsprechend der folgenden zwei Ebenen:

- *Konzeptionelle Ebene:* Das Prozessmetamodell unterstützt die Definition von taskspezifischen Qualitätsmetriken und -zielen. Qualitätsmetriken beschreiben (potenziell) aggregierte (Qualitäts-)Parameter, wie zum Beispiel den prozentualen Anteil erfüllter Anforderungen oder den prozentualen Abdeckungsgrad an Anforderungen

der oberen Abstraktionsebene. Die Definition einer Qualitätsmetrik muss für jede Instanz eines Work Products anwendbar sein, für den diese definiert ist. Ein Beispiel wäre eine einfache Qualitätsmetrik, welche den Prozentsatz bereits erfüllter Anforderungen in einer Verifikationsphase berechnet. Diese Metrik ist so zu definieren, dass sie gegen jedes konkrete Anforderungsmodell auswertbar ist. Eine Definition basierend auf Konzepten des Produktmetamodells ermöglicht dies.

- *Instanz Ebene:* Die formale Definition der Qualitätsmetriken basierend auf dem Produktmetamodell ist nur eine Seite der Medaille bei der Integration zwischen Prozess- und Produktsicht. Ferner ist für ein Work Product in einem Prozess zu ermitteln, welchen konkreten Design Artefakten sie im Kontext einer Prozessinstanziierung (z.B. ein Projektplan) entsprechen. Dies lässt sich erst nach einer Prozessinstanziierung bestimmen, da derselbe Prozess mit denselben Work Products in der Produktentwicklung mehrfach angewandt wird. Schafft man es für jedes Work Product eines instanziierten Prozesses zur Laufzeit genau die gültigen Design Artefakte zu bestimmen, ist eine automatische Auswertung aller definierten Qualitätsmetriken und –ziele auf Basis der aktuellen Eigenschaften der Design Artefakte möglich.

Die Integration von Produkt- und Prozessmetamodell wird im Folgenden auf beiden Ebenen behandelt.

Konzeptionelle Ebene

Dieser Abschnitt beschreibt das Vorgehen zur Definition von Qualitätsmetriken basierend auf dem Produktmetamodell. Die Integration wird als konzeptionell bezeichnet, da sie lediglich die Metamodelle und nicht deren Instanzen betrachtet.

Die konzeptionelle Integration im Prozessmetamodell betrifft die Konzepte zur Repräsentation von Ein- und Ausgang von Entwicklungsschritten. Die in SPEM existierenden Konzepte zur Beschreibung der Work Products wurden bereits in der Version 1.1 als unzureichend definiert (siehe (Gonzales-Perez, 2005)). Ein Kritikpunkt, der sich auch in der aktuellen Version nicht geändert hat. Das zentrale Konzept zur Beschreibung der Work Products von Prozessen in SPEM ist `WorkProductDefinition`, welche durch `WorkProductKind` typisierbar sind. Die SPEM-Spezifikation führt dabei bereits einige Typen ein. Diese gehen jedoch nicht über den Typ `Artifact`, `Deliverable` oder `Outcome` (siehe Abbildung 33) hinaus. So umfasst `Artifact` laut Dokumentation sowohl Modelle als auch einzelne Modellelemente, was einer sehr abstrakten und semantisch schwachen Typisierung entspricht, da dies nahezu alle Entwicklungsergebnisse umfasst.

Das Core-Produktmetamodell wird an dieser Stelle konkreter, führt weitere Modelle (zum Beispiel Anforderungsmodell) und Modellelemente (zum Beispiel Komponente) ein und erlaubt domänenspezifische Spezialisierungen. Alle Modell und Modellelemente im Produktmetamodell können als Konkretisierung des `Artifact` Konzepts in SPEM (siehe Abbildung 33) betrachtet werden. Beide Sichten ergänzen sich demnach optimal. Ziel der Arbeit ist eine lose Kopplung (siehe **Anforderung 3**) mit möglichst

minimalen Abhängigkeiten zwischen den Metamodellen selbst, weshalb die Verbindung in Abbildung 33 gestrichelt dargestellt ist.

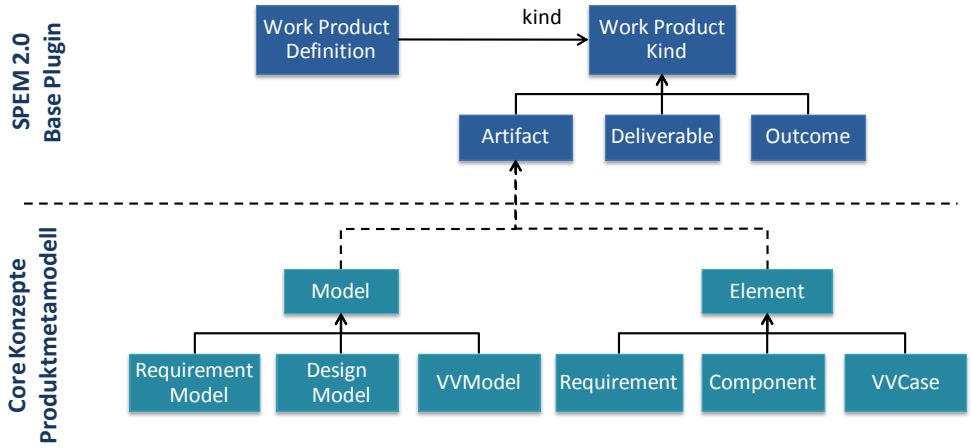

Abbildung 33: Überschneidende Konzepte aus Prozess- und Produktmetamodell

Um eine lose Kopplung zu erreichen, wird ein abfragebasierter Ansatz (z.B. die Object Constraint Language (OCL, 2010) (siehe Kapitel 4) als Mittel zur formalen Definition von Work Products gewählt. Die Bedeutung dessen erläutert Abbildung 34. Das Beispiel zeigt ein Work Product „Anforderungsmodell“ und eine darauf definierte Qualitätsmetrik „Anzahl erfüllte Anforderungen“. Zur Laufzeit ist mit Hilfe einer definierten Abfrage zu identifizieren, welche `SystemRequirement` Instanzen zum Work Product „Anforderungsmodell“ gehören. Jede definierte Abfrage geht von einem oder mehreren Kontextobjekten aus. Für ein Work Product ist dies das `Component` Konzept im Produktmetamodell, denn eine Prozessdefinition wird für ein System, bzw. für eine Systemkomponente instanziiert (siehe Integration auf Instanzebene).

Betrachten wir wieder das in Abbildung 34 dargestellte Beispiel. Das Work Product (`WorkProductDefinition`) „Anforderungsmodell“ wird als eine Anforderungsmenge definiert, welche über eine `Satisfy` Relation mit der `Component` Instanz verknüpft ist, auf die sich die Prozessinstanz bezieht. Ergebnis ist eine Menge an `SystemRequirement` Instanzen. Grundsätzlich lässt sich jede Art von Abfrage mit beliebigen Rückgabetypen für ein Work Product definieren, die von der Definition des `Component` Konzepts und seiner Relationen und Attribute startet. Eine `MeasureDefinition` arbeitet auf dem Output des Work Products auf das sie sich bezieht. Für die `MeasureDefinition` „Anzahl erfüllte Anforderungen“ sind dies eine Menge an `SystemRequirement`, was sich aus der formalen Definition des Work Products ableiten lässt. Entsprechend beginnen die Abfragen eines `BaseMeasure` Objekts mit der konzeptionellen Beschreibung (Relationen, Attribute) von `SystemRequirement`. In dem Beispiel müsste die Qualitätsmetrik die Erfüllungsgrade für jede Anforderung berechnen und dann die Anforderungen zählen. Ergebnis ist eine Zahl (z.B. 27).

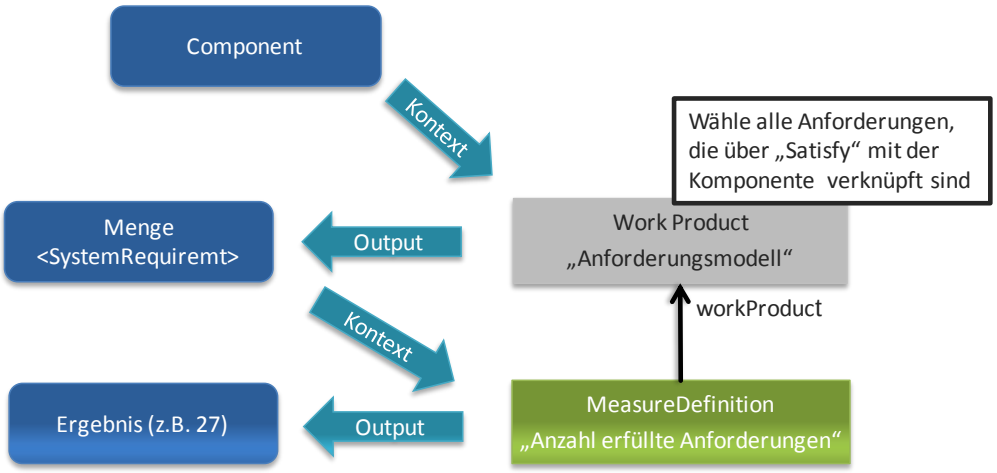

Abbildung 34: Beispiel konzeptionelle Integration

Eine formale Definition eines Work Products geht nicht zwangsläufig von dem Konzept `Component` aus. Abbildung 35 beschreibt dies beispielhaft. Handelt es sich um ein Work Product, welches ein anderes bereits formal vorliegendes Work Product mittels der in SPEM vorhandenen `Extends` Relation erweitert, dann bezieht sich die formale Definition auf den Output des erweiterten Work Products. Abbildung 35 zeigt, wie ein Work Product „Sicherheitsanforderungen“ das Work Product „Anforderungsmodell“ spezialisiert. Die Abfrage für „Sicherheitsanforderungen“ startet demnach von dem Output von „Anforderungsmodell“. Im Beispiel wird eine Menge an `SystemRequirements` gefiltert, hier entsprechend der Kategorisierung „Sicherheitsanforderungen“. Das Nutzen dieses bereits existierenden Extension Mechanismus erlaubt beliebige Verfeinerungen von Work Products und bleibt bei deren Formalisierung auf einem simplen Level.

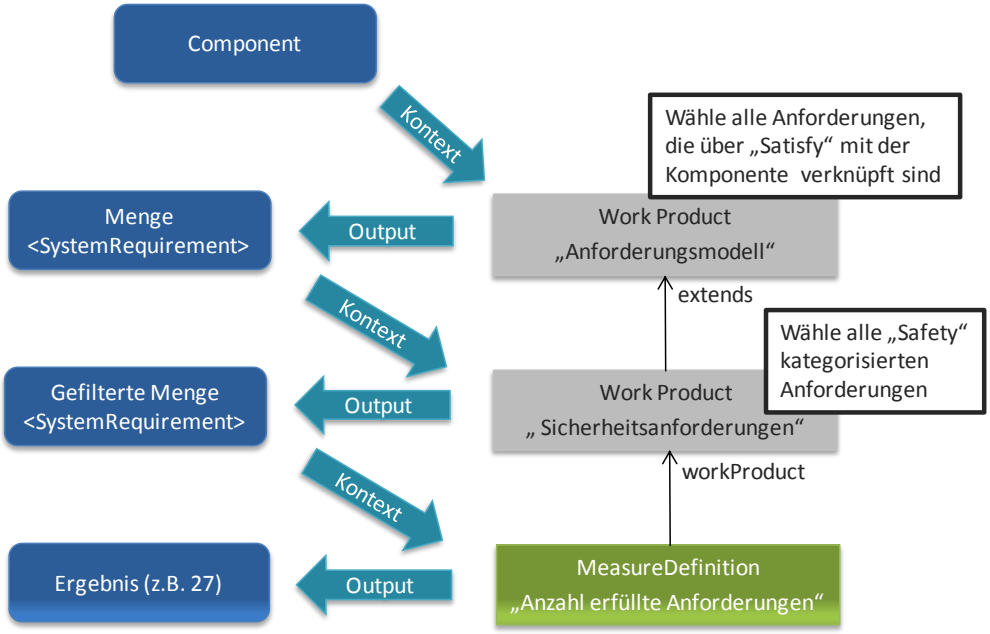

Abbildung 35: Beispiel mit Work Product Spezialisierung

Work Products und Qualitätsmetriken sind nun auf Basis des Produktmetamodells definierbar, es fehlt jedoch noch die Instanziierung von Prozessen für eine konkrete Komponente, um während der operativen Durchführung von Projekten eindeutig zu identifizieren, gegen welchen konkreten Kontext (definiert in Form einer Komponente) die definierten Abfragen auszuwerten sind.

Instanz Ebene

Zur Integration auf Instanz Ebene werden die vorliegenden Prozess- und Qualitätsmetrikdefinitionen zunächst für ein Projekt instanziiert, in dem sie für eine oder mehrere Komponenten verwendet werden. In Abbildung 36 ist dies beispielhaft dargestellt, ein Prozess wird für unterschiedlich sicherheitskritische (ASIL – Automotive Safety Integrity Level) Komponenten instanziiert. Ergebnis ist eine kalibrierte Prozess- und Qualitätsmetrikdefinition, die genau die durch einen Sicherheitsstandard geforderten Prozessschritte enthält (genauere Erläuterungen hierzu finden sich in Abschnitt 5.2).

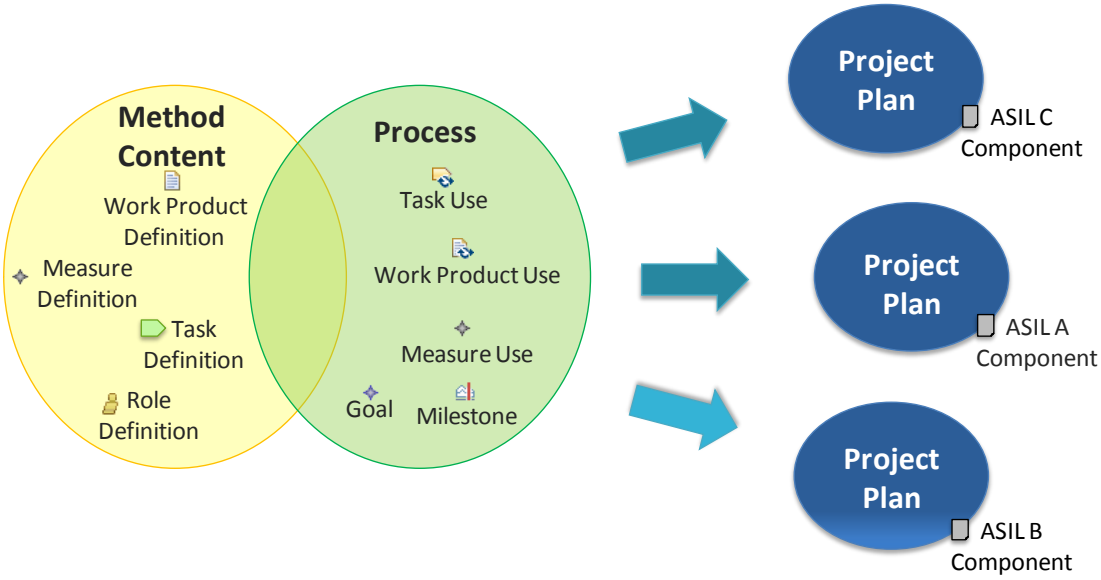

Abbildung 36: Prozessinstanziierung

Basierend auf der verknüpften Komponente und der bereits beschriebenen Definition von Work Products und Qualitätsmetriken, lassen sich die Design Artefakte dynamisch unter Verwendung des Produktmetamodells bestimmen.

Abbildung 37 zeigt eine beispielhafte Instanziierung der Modelle für die Bewertung einer Task, in diesem Fall die Verifikation des System-Designs gegen die Anforderungen mittels Simulation. *R* steht dabei für `Requirement`, *C* für `Component` und *V* für `VVCase`. Die Qualitätsmetrik berechnet den Erfüllungsgrad von Sicherheitsanforderungen als Verhältnis erfüllter Anforderungen zu allen definierten Anforderungen einer Systemarchitektur, beziehungsweise einer einzelnen Komponente der Architektur. Dabei wird der Erfüllungsgrad im Kontext dieser Task aus Sicht der Verifikationsmethode Simulation betrachtet. Das heißt, eine Sicherheitsanforderung ist im Kontext dieser Task nur dann erfüllt, wenn alle definierten Simulationsläufe die gewünschten Ergebnisse

liefern. Sind zum Beispiel weitere Verifikationsmethoden im Prozess definiert, werden diese in gesonderten Tasks betrachtet. Im Beispiel ist der Prozess (und damit implizit auch jeder einzelne Prozessschritt) für die gesamten Systemarchitektur definiert. Geht man von einer formalen Definition des Work Products „Technische Sicherheitsanforderungen“ entsprechend des dargestellten Beispiels in Abbildung 37 aus, fließen alle Sicherheitsanforderungen der Systemarchitektur in die Bewertung mit ein. Entsprechend der Kategorisierung der Anforderungen wird die Berechnung der Erfüllungsgrade durchgeführt.

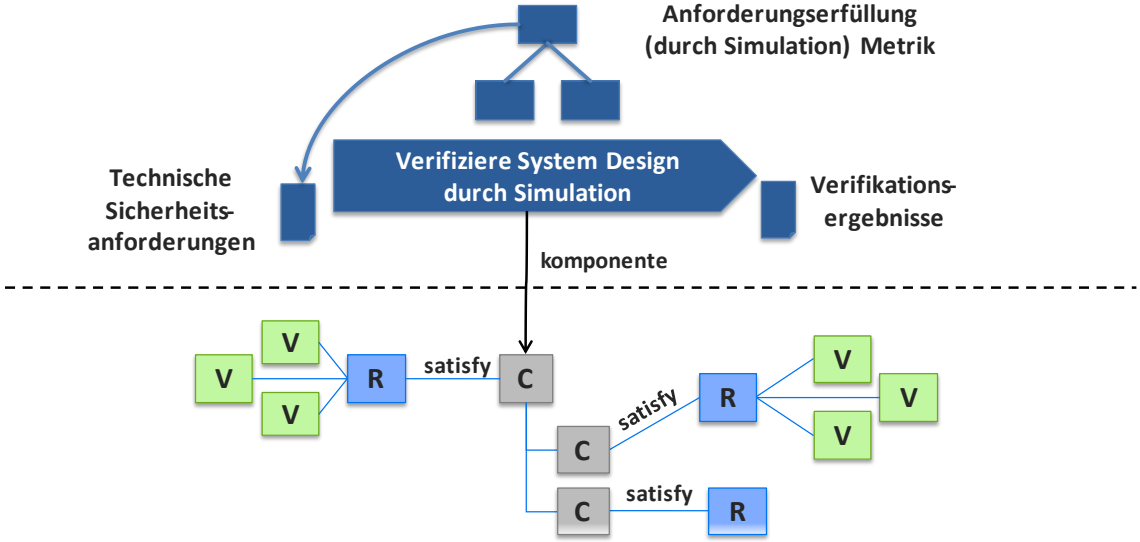

Abbildung 37: Verknüpfung Work Product (Prozess) und Komponente (Produkt)

Zusammenfassung – Durch die in diesem Kapitel beschriebene Integration zwischen Prozess- und Produktsicht ist es nun möglich eine prozessorientierte Qualitätsdefinition durchzuführen und gegen konkrete Design Artefakte auszuwerten. Dies erlaubt ein modellbasiertes und prozessorientiertes Produktqualitätsmonitoring. Da die Anwendung aber immer mit Kalibrierungsaufwand verbunden ist, führt der folgende Abschnitt ein Vorgehensmodell ein, unter dessen Anleitung Unternehmen das Qualitätsmodellierungsframework zum einen auf ihren unternehmensinternen Entwicklungsprozess kalibrieren und zum anderen während der Projektplanung und -steuerung anwenden.

3.3 Vorgehensmodell

Das entwickelte Vorgehensmodell orientiert sich an dem ISO 15939 (ISO15939, 2007) Standard, in dem der Bewertungsprozess in drei Hauptaktivitäten unterteilt ist: Planung, Durchführung und Evaluierung. Das in dieser Arbeit entwickelte Vorgehensmodell fokussiert die Phasen Planung und Durchführung und gestaltet diese mit Hilfe des Qualitätsmodellierungsframeworks im Detail aus. Die Evaluierungs- und Optimierungsphase ist nicht Fokus dieser Arbeit.

Abbildung 38 stellt die in dieser Arbeit entwickelte Ausprägung eines generellen Bewertungsprozesses hin zu einer prozessorientierten Qualitätsbewertung dar. Die folgen-

den Unterabschnitte widmen sich sowohl der Planungs- als auch der Durchführungsphase und erläutern sie im Detail.

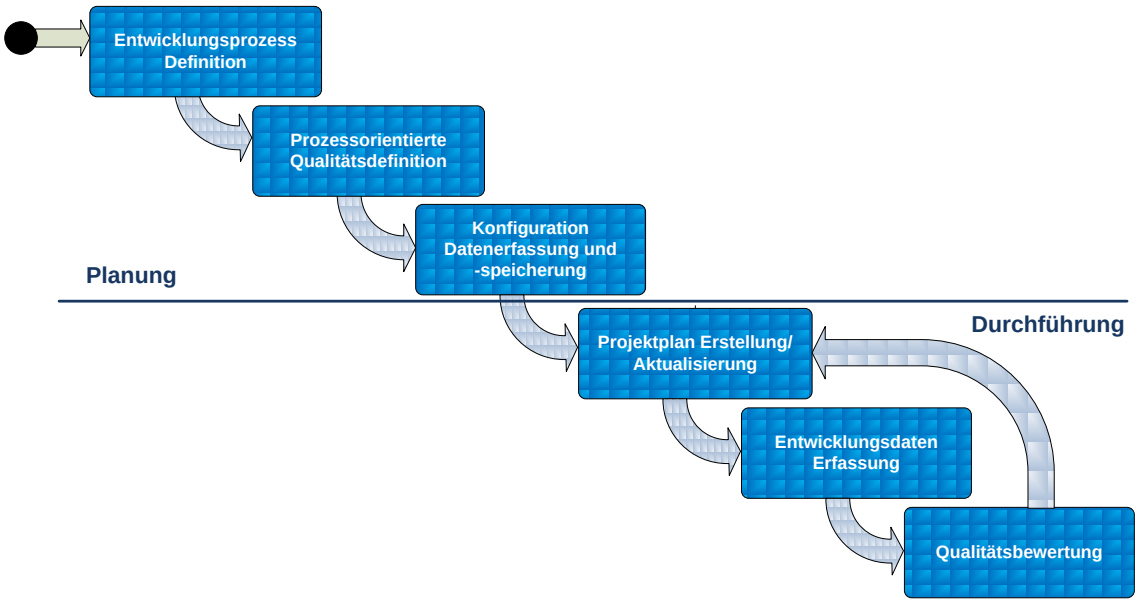

Abbildung 38: Vorgehensbeschreibung für ein entwicklungsbegleitendes Produktqualitätsmonitoring

3.3.1 Planungsphase

Die Planungsphase besteht aus drei Schritten, die sich jeweils in weitere Unterschritte untergliedern lassen.

1. Entwicklungsprozessdefinition

Der erste Schritt des Vorgehensmodells sieht die Definition des Entwicklungsprozesses mit Hilfe eines auf dem Prozessmetamodell basierendem Modellierungswerkzeugs (siehe Kapitel 4, z.B. eine Erweiterung des Eclipse Process Framework[8]) vor und umfasst die typischen Arbeitsschritte, die bereits durch SPEM 2.0 unterstützt werden.

a. Definiere Tasks und Work Products

Input: Verwendete Entwicklungsmethoden, Beschreibung von Work Products

Output: Methodenbibliothek mit Tasks und Work Products

Beschreibung: Es sind zunächst alle Bausteine des eigenen Entwicklungsprozesses zu definieren. Hierzu gehören zum einen alle Tasks und zum anderen alle Work Products des Entwicklungsprozesses.

b. Definiere Prozesse

Input: Methodenbibliothek, Prozessbeschreibungen

Output: Prozessmodell im Prozessmetamodell Format

[8] http://www.eclipse.org/epf/

Beschreibung: Basierend auf existierender Prozessdokumentation und der definierten Methodenbibliothek werden ein oder mehrere Prozesse erstellt.

2. Prozessorientierte Qualitätsdefinition

Basierend auf der vorliegenden Definition von Methoden und Prozessen erfolgt im zweiten Schritt die prozessorientierte Qualitätsdefinition. Dieser Schritt besteht aus folgenden drei Unterschritten:

a. Formalisiere Work Products in Prozessmetamodell

Input: Definition Produktmetamodell, Prozessdefinitionen

Output: Formal definierte Work Products

Beschreibung: Jedes Work Product mit ein oder mehreren definierten Qualitätsmetriken wird mittels einer Abfrage basierend auf dem Produktmetamodell beschrieben.

b. Definiere Qualitätsmetriken

Input: Methodenbibliothek, existierende Metrik Dokumentation, Standards

Output: Formalisierte Qualitätsmetriken

Beschreibung: Unter Verwendung eines Prozessmodellierungswerkzeugs wird die Methodenbibliothek um Qualitätsmetriken entsprechend der Beschreibung in Kapitel 3.2.2 erweitert. Grundlage hierfür sind z.B. bereits im Unternehmen oder in Standards formulierte Metriken. Deckt das Produktmetamodell nicht alle für eine Metrik notwendigen Informationen ab, sind zunächst Metamodellerweiterungen einzuführen (siehe 3.b.), bevor dieser Schritt abgeschlossen werden kann.

c. Definiere Qualitätsziele

Input: Methodenbibliothek inkl. Qualitätsmetriken, Prozessmetamodell basierte Prozessdefinition

Output: Prozess- und Meilensteinziele

Beschreibung: Der bereits vorher definierte Prozess beinhaltet alle Qualitätsmetriken seiner einzelnen Schritte. Es besteht die Möglichkeit, diese Metriken um prozessspezifische Zieldefinitionen zu erweitern, was z.B. in iterativen Prozessen denkbar ist, in denen derselbe Prozessschritt mehrfach durchgeführt wird. Meilensteinziele sind ebenfalls definierbar.

3. Konfiguration Datenerfassung und -speicherung

Als Ergebnis der ersten beiden Schritte liegt eine formale Definition von Prozess- und Qualitätsdefinition vor. Aus diesen Informationen lassen sich die notwendigen Produktdaten für die Qualitätsbewertung ableiten. Die folgenden zwei Unterschritte dienen zur Konfiguration der Datenerfassung und -speicherung:

a. Definition von Transformationen

Input: Datenformate Entwicklungswerkzeuge und –datenbanken, (erweitertes) Produktmetamodell

Output: Transformationen

Beschreibung: Die benötigten Daten zur Qualitätsbewertung sind aus den proprietären Datenformaten der Entwicklungswerkzeuge und –datenbanken in das definierte Metamodell zu transformieren. Entsprechende Modelltransformationen sind zu definieren und zu implementieren. Sollte es nicht möglich sein, notwendige Daten auf das definierte Core-Produktmetamodell abzubilden, sind Metamodellerweiterungen durchzuführen (siehe Schritt 3.b.). Diese Modelltransformationen sind als Adapter in das verwendete Auswertungswerkzeug zur integrieren.

b. Erweiterung Produktmetamodell (optional)

Input: Prozessmodell, prozessorientierte Qualitätsdefinition, Transformationen

Output: Produktmetamodellerweiterungen

Beschreibung: Dieser optionale Schritt wird durchgeführt, sollte eine Repräsentation notwendiger Produktdaten zur Berechnung der Qualitätsmetriken allein mit dem Core-Produktmetamodell nicht möglich sein. In diesem Fall dienen weitere Partialmodelle als Erweiterung des Core-Produktmetamodells. Je nach dessen Art sind entweder die Anforderungs-, Design oder V&V Konzepte durch Einführung von Subklassen zu erweitern. Auch ein nachträgliches hinzufügen weiterer Anforderungsverfolgungskonzepte ist denkbar. Jedes Partialmodell ist entsprechend der verwendeten Metamodellierungssprache zu implementieren (siehe z.B. Kapitel 4).

3.3.2 Ausführungsphase

Ist das Qualitätsmodellierungsframework durch die Prozess- und Qualitätsdefinition und die Mechanismen zur Datenerfassung und –speicherung konfiguriert, ist die Grundlage für die operative Durchführung einer entwicklungsbegleitenden Qualitätsbewertung geschaffen. Das Vorgehen zur Anwendung der Qualitätsbewertung für ein konkretes Projekt gliedert sich in die folgenden drei Schritte:

1. Projektplanerstellung

Input: Prozessmodell, Design Modell

Output: Projektplan

Beschreibung: Basierend auf einer oder mehreren Prozessbeschreibungen wird ein Projektplan generiert. Projektpläne sind für jede Komponente eines Design Modells erstellbar, so dass spezifisch für jede Komponente auch abweichende Projektpläne erzeugbar sind. Ergebnis ist eine Projektplanvorlage in dem sich jeder einzelne Schritt auf eine Komponente bezieht. Die erzeugten Templates lassen sich mit einem Projekt-

managementwerkzeug öffnen und mit Start- und Endterminen weiter vervollständigen (z.B. mit der im Prototyp implementierten MS Project Anbindung).

2. Entwicklungsdaten Erfassung

Input: Modelltransformationen

Output: Aktueller Produktentwicklungsstand

Beschreibung: Unter Verwendung definierter Modelltransformationen wird der aktuelle Entwicklungsstand regelmäßig automatisiert erfasst.

3. Qualitätsbewertung

Input: Entwicklungsdaten, prozessorientierte Qualitätsdefinition annotiert an Prozessinstanz (d.h. Projektplan)

Output: Ergebnisse Qualitätsbewertung

Beschreibung: Definierte Qualitätsmetriken und –ziele werden ausgewertet und mit Prozess- und Meilensteinzielen abgeglichen. Alle berechneten Ergebnisse werden Projektmanagern und Entwicklern dargestellt, um auf mögliche Probleme hinzuweisen. Werden während der Projektüberwachung Probleme lokalisiert, die zu nicht einzuhaltenden Projektzielen und –terminen führen, ist ein Schritt zurück (siehe Abbildung 38 Pfeil von Qualitätsfortschrittsbewertung hin zu Projektplanerstellung) in die Phase der „Projektplanung“ erforderlich, um Termine und Ziele zu aktualisieren.

3.4 Zusammenfassung

Ziel dieses Kapitels war die Entwicklung einer innovativen Lösung für eine entwicklungsbegleitende Produktqualitätsbewertung. Die Lösung adressiert den im Stand der Technik Kapitel identifizierten Handlungsbedarf und die zu Beginn dieses Kapitels daraus abgeleiteten Anforderungen wie folgt:

Handlungsbedarf – Prozessorientierte Qualitätsbewertung:

Um die notwendige prozessorientierte Qualitätsbewertung zu ermöglichen, galt es zunächst eine formale Beschreibung von Prozessen, Qualitätsmetriken und -zielen innerhalb des Qualitätsmodellierungsframeworks zu unterstützen. Die aufgestellten Anforderungen wurden wie folgt adressiert:

- ***Anforderung 1 –*** *Konzepte zur Prozessdefinition* - Die Verwendung von SPEM 2.0 als Prozessmetamodell (3.2.2) innerhalb des Qualitätsmodellierungsframeworks erlaubt den Zugriff auf alle notwendigen Konzepte zur Prozessmodellierung.
- ***Anforderung 2 –*** *Konzepte zur Definition von Qualitätsmetriken und –zielen* – Das verwendete Prozessmetamodell SPEM 2.0 wurde durch formale Measurement Konzepte zur Definition von Qualitätsmetriken (3.2.2) erweitert. Die Qualitätsmessung passt sich der Prozessdefinition des jeweiligen Projekts an und er-

laubt darüber hinaus die Definition prozessspezifischer Ziele für die definierten Qualitätsmetriken.

- ***Anforderung 3 –*** *Konzept zur (losen) Integration von Prozess- und Produktsicht* - Um Produkt- und Prozesssicht miteinander zu verknüpfen wurde als ein zentraler Beitrag dieser Arbeit eine Integration auf Metamodellebene geschaffen (3.2.3). Integrationspunkte stellen die Work Products des Prozessmodells dar. Ausgehend von Work Products besteht die Möglichkeit beliebige Verknüpfungsregeln zu definieren. Korrespondierende Design Artefakte im Produktmetamodell werden dynamisch, ohne Notwendigkeit einer direkten Verknüpfung der Metamodelle, den Work Products in einer Prozessinstanz zugeordnet.

Handlungsbedarf – Systematische Anpassung auf einzelne Projekte

Die Arbeit beantwortet Anforderungen bezüglich einer systematischen und effizienten Anpassung von Qualitätsdefinitionen auf einen konkreten Projektkontext wie folgt:

- ***Anforderung 4 –*** *Effiziente Kalibrierung der Qualitätsmessung auf einzelne Projekte* – Das Konzept sieht die Instanziierung von Prozessen für konkrete Projekte vor, die in Vorlagen für Projektpläne resultieren. Diese Kalibrierung führt gleichzeitig zu einer projektspezifischen Qualitätsmessung, die nur Qualitätsmetriken berücksichtigt, die auch an verwendeten Tasks annotiert sind.

- ***Anforderung 5 –*** *Gewährleistung einer stringenten Anforderungsorientierung für ein zielorientiertes Messen* – Das Produktmetamodell erlaubt die Erfassung aller im Projekt gültigen Anforderungen, deren Allokation zu Komponenten des Systems und der Erfassung aller V&V Aktivitäten (3.2.1) und ermöglicht so eine Bewertung von Anforderungsqualität und –erfüllung. Die Anforderungsorientierung sorgt dafür, dass nur die Qualitätscharakteristika berücksichtigt werden, die auch in einem Projekt gefordert sind. Das Produktmetamodell basiert zu einem Großteil auf etablierten Vorarbeiten wie SysML und EAST-ADL2 und wurde entsprechend den Anforderungen punktuell erweitert.

Handlungsbedarf – Einheitliche Sicht auf qualitätsrelevante Daten:

Das Core-Produktmetamodell erlaubt die Integration aller qualitätsrelevanten Daten von der Qualitätsdefinition in Form von Anforderungen bis hin zu deren Verifikation und Validierung. Die Anforderungen an das Konzept wurden wie folgt umgesetzt:

- ***Anforderung 6 –*** *Konzepte zur Repräsentation von Anforderungen* - Das im Produktmetamodell enthaltende Anforderungsmetamodell liefert alle notwendigen Konzepte zur Abbildung funktionaler wie nicht funktionaler Anforderungen. Eine integrierte Abbildung von Anforderungen, egal welcher Art und welcher Formalisierung erlaubt die Berücksichtigung aller projektspezifischer Qualitätscharakteristika. Ein entsprechendes Metamodell existierte so bisher nicht.

- ***Anforderung 7 –*** *Konzepte für die Anforderungsverfolgung* - Das im Produktmetamodell enthaltende Anforderungsverfolgungsmetamodell enthält die elementaren Konzepte zur Nachvollziehbarkeit von Anforderungsverfeinerungen, Alloka-

tion sowie Verifikation und Validierung. Die Konzepte basieren auf existierenden Vorarbeiten wie SysML und EAST-ADL2.

- ***Anforderung 8 –*** *Konzepte zur Abbildung von Analysen und Verifikationsergebnissen* - Das im Produktmetamodell enthaltende V&V Metamodell erweitert das V&V Paket aus EAST-ADL2 und erlaubt die Zusammenführung aller V&V Fälle und deren Ergebnisse. Die Verknüpfungen mit definierten Anforderungen erlauben ein genaues Tracking der Qualität und Erfüllung eben dieser.

Handlungsbedarf – Domänenübergreifende Betrachtung

Das entwickelte Qualitätsmodellierungsframework löst die Anforderungen hinsichtlich des Handlungsbedarfs für eine domänenübergreifende Betrachtung von Qualität wie folgt:

- ***Anforderung 9 –*** *Berücksichtigung verschiedener Partialmodelle* - Das Qualitätsmodellierungsframework erlaubt die Abbildung beliebiger Produktpartialmodelle auf das Core-Produktmetamodell. Dies ist im Wesentlichen Aufgabe der Modelltransformationen.

- ***Anforderung 10 –*** *Erweiterungsmechanismen für verschiedene Partialmodelle* - Sollte das Core-Produktmetamodell einmal nicht ausdrucksstark genug sein, können auf einfache Weise durch Erweiterungen des Core-Produktmetamodells neue Konzepte eingeführt werden.

Handlungsbedarf –Werkzeugunterstützung

Das Konzeptkapitel legt die Grundlage für die Adressierung dieses Handlungsbedarfs. Final wird diese jedoch erst mit dem folgenden Kapitel gelöst:

- ***Anforderung 11 –*** *Formale Modellgrundlage* – Die definierten Metamodelle bilden eine Vorlage zur Implementierung mittels einer geeigneten Metamodellsprache.

- ***Anforderung 12 –*** *Automatische entwicklungsbegleitende Datenerfassung, -integration und –analyse* – Implementierte Transformationen erlauben eine automatische Erfassung von Entwicklungsdaten und deren Überführung in das einheitliche Datenformat. Die formale Beschreibung der Qualitätsmetriken erlaubt eine automatisierte Analyse dieser Daten. Beides ist je nach Bedarf zu beliebigen Zeitpunkten während der Entwicklung durchführbar.

- ***Anforderung 13 –*** *Anbindung an unternehmensinterne IT* – Der Metamodellbasierte Ansatz erlaubt die Anbindung prinzipiell beliebiger Werkzeuge mittels Modeltransformationen. Dies gilt sowohl für die Definition von Prozessen und Projektplänen als auch für jedwede Art von Entwicklungsdaten.

Das in diesem Kapitel entwickelte Konzept legt die Grundlage für die Werkzeugunterstützung, die im folgenden Kapitel prototypisch umgesetzt wird.

4 Prototypische Umsetzung

Wie bereits im Stand der Technik angedeutet und auch im Konzeptkapitel 3 erwähnt, ist eine Qualitätsbewertungsmethodik ohne entsprechende Werkzeugunterstützung nicht praktikabel umsetzbar. Dieses Kapitel beschreibt die Umsetzung des entwickelten Lösungskonzepts und adressiert damit die letzten bisher nicht direkt adressierten Anforderungen hinsichtlich notwendiger Werkzeugunterstützung und schließt diese Schwachstelle. Der Prototyp dient der Evaluierung (siehe Kapitel 5) der Methodik anhand von zwei Anwendungsbeispielen und baut auf einer Version aus (Hausmann, 2008) auf. Abschnitt 4.1 beschreibt die Architektur des Prototyps und Abschnitt 4.2 die Umsetzung der einzelnen Komponenten entlang des entwickelten Vorgehensmodells.

4.1 Architektur

Dieser Abschnitt beschreibt die Architektur des Prototyps zur Umsetzung der entwickelten Methodik aus Kapitel 3. Abbildung 39 visualisiert die Architektur, welche sich in mehrere Teilkomponenten aufgliedert. Allgemein unterteilt sich die Architektur in vier horizontal dargestellte Schichten. Die untere Schicht „Modellierungs- und Analysewerkzeuge“ umfasst alle in der Entwicklung verwendeten Werkzeuge zur Beschreibung von Produkt-, Prozess- und Projektartefakten. Die obere Schicht „Services“ enthält alle implementierten Komponenten für das prozessorientierte Produktqualitätsmonitoring. Die Schnittstelle zwischen den Komponenten der genannten Schichten stellen die Schichten „Metamodelle“ und „Transformationen“ dar. Die Implementierungen der Metamodelle zur Abbildung von Prozessmodellen, Projektplänen und Produktdaten sind zentraler Bestandteil des Prototyps und wurden mit Hilfe des Ecore Formats definiert. Basierend auf diesen Modellen wurde mit Hilfe des Eclipse Modelling Frameworks[9] Quellcode zur Abbildung der Daten generiert (siehe die drei API’s in der Schicht „Metamodelle“ in Abbildung 39). Auf eine detaillierte Beschreibung der Modelle wird an dieser Stelle verzichtet. Diese Modelle stellen die Implementierung der **Anforderungen 1 und 2** hinsichtlich der Konzepte zur Prozessmodellierung und der **Anforderungen 6 bis 8** zur Abbildung der Produktdaten dar. Die Verwendung von Ecore erfüllt **Anforderung 12** hinsichtlich einer notwendigen formalen Modellgrundlage. Für jedwedes Werkzeug, sei es EPF, MS Project oder andere Entwicklungswerkzeuge existieren Transformationen, um die proprietären Daten in das einheitliche Metamodellformat zu übertragen.

[9] http://www.eclipse.org/modeling/emf/

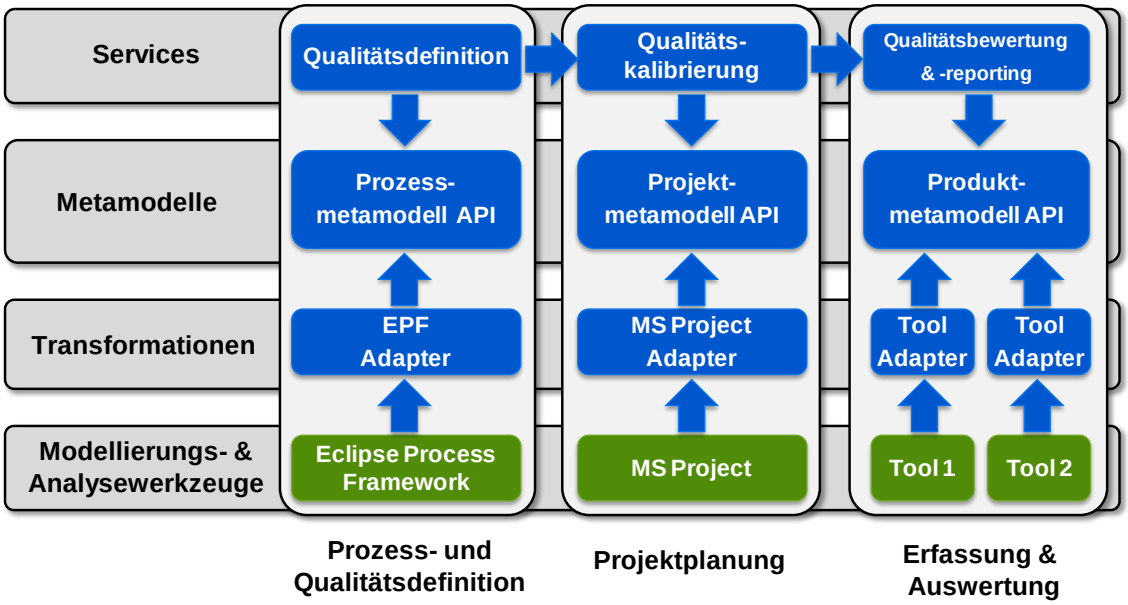

Abbildung 39: Grobarchitektur

Neben der horizontalen Unterteilung der Architektur in vier Schichten, beschreibt Abbildung 39 auch eine vertikale Unterteilung in die drei Hauptfunktionalitäten des Prototyps. Diese Hauptfunktionalitäten werden in den folgenden Abschnitten beschrieben und an dieser Stelle eingeführt:

Prozess- und Qualitätsdefinition – Qualitätsbewertungen werden zielorientiert basierend auf einer Prozess- und Qualitätsdefinition durchgeführt. Zu diesem Zweck integriert der Prototyp das Werkzeug EPF Composer und erweitert dieses um die beschriebenen Measurement Konzepte und der notwendigen Funktionalitäten zur formalen Definition von Qualitätsmetriken und Work Products basierend auf dem Produktmetamodell. Eine detaillierte Beschreibung der Umsetzung der Funktionalitäten findet sich in Abschnitt 4.2.1 und 4.2.2. Die Komponenten stellen die Werkzeugimplementierung für das Prozessmetamodell und damit einen weiteren Teil der Realisierung von **Anforderung 1 und 2** dar. Darüber hinaus erlaubt es die konzeptionelle Integration zwischen Produkt- und Prozessmetamodell und erfüllt einen Teil von **Anforderung 3**.

Projektplanung – Die Komponente zur Unterstützung von Projektplanung und Überwachung bindet zum einen exemplarisch das Projektmanagementwerkzeug MS Project an, um die Qualitätsbewertungen direkt bei der Prozessplanung und -steuerung zu berücksichtigen. Ebenfalls wurden Funktionalitäten implementiert, die aus definierten Prozessen initiale Projektpläne erzeugen. Dies erlaubt die Wiederverwendung einer Definitionen für eine prozessorientierte Qualitätsbewertung in mehreren Projekten, ohne dabei aber die Kalibrierung zu vernachlässigen, da nur genau die Teile instanziiert werden, die im Projekt notwendig sind. Die Komponenten realisieren damit **Anforderung 4** hinsichtlich der Kalibrierung der Qualitätsdefinition auf einzelne Projekte und **Anforderung 3**, da die Instanziierung eindeutig festlegt, gegen welchen Teil des Systems die Qualitätsbewertungen ausgeführt werden sollen. Eine detaillierte Beschreibung dieser Komponenten findet sich in Abschnitt 4.2.3.

Erfassung & Auswertung – Neben der Definition eines projektspezifischen Plans zur Qualitätsmessung, ermöglicht der Prototyp ebenfalls die Qualitätsauswertung und -visualisierung basierend auf dem aktuellen Entwicklungsstand. Die entsprechende Komponente zur „Qualitätsbewertung“ implementiert **Anforderung 5**, da sie Auswertungen einer anforderungsorientierten Qualitätsbewertung umsetzt. Ferner implementiert sie teilweise **Anforderung 12**, da sie die Automatisierung der Datenanalyse übernimmt. Da selbst der Prototyp auf verschiedene Anwendungsbereiche angewandt wird (siehe Kapitel 5) und prinzipiell offen für weitere Anwendungsbeispiele bleiben soll, wurden Schnittstellen zur Anbindung beliebiger Entwicklungswerkzeuge und –datenbanken geschaffen. Lediglich eine elementare Voraussetzung ist das Einhalten der Core-Produktmetamodelldefinition. Der Prototyp speichert Qualitätsbewertungsergebnisse und deren Verlauf über die Zeit. Damit adressiert der Prototyp **Anforderung 9, 10 und 13**, da sie für verschiedene Domänen und Partialmodelle unabhängig von der konkret verwendeten unternehmensinternen IT kalibrierbar ist (siehe auch (Hausmann, 2009)). Darüber hinaus adressiert die Komponente **Anforderung 12**, da diese Komponente eine entwicklungsbegleitende Erfassung der Produktdaten möglich macht. Abschnitt 4.2.4. beschreibt Datenerfassung und -auswertung im Detail.

4.2 Implementierung

Die folgenden Abschnitte beschreiben Umsetzung und Anwendung der Komponenten der in Abschnitt 4.1 eingeführten Grobarchitektur entlang des in dieser Arbeit entwickelten Qualitätsbewertungsprozesses. Lediglich die Beschreibung der Metamodell Implementierung, sowie die Erfassung der Entwicklungsdaten selbst („Ausführungsphase“ – Schritt 2) wird an dieser Stelle nicht weiter detailliert. Die Umsetzung erfolgte mit Hilfe der Programmiersprache Java und der Eclipse „Rich Client Plattform“ (kurz RCP). Die Eclipse RCP wurde mit der Eclipse Version 3.0 veröffentlicht. Die RCP baut auf einer Reihe verschiedener Plugins auf, welche durch den Entwickler beliebig kombinierbar sind. Von der Zusammenstellung der minimalen RCP Plugins, die unter anderem Bibliotheken wie das Standard Widget Toolkit oder JFace für die Gestaltung von Benutzeroberflächen enthalten, sind weitere Plugins, wie Navigatoren oder die gesamte Eclipse IDE verwendbar. RCP Entwickler können daher aus einem breiten Fundus wiederverwendbarer Funktionalität schöpfen. Auf Grund der Verwendung der Eclipse RCP ist auch der Prototyp in verschiedene Plugins organisiert. Dies fördert die komponentenorientierte Entwicklung und erlaubt die Zusammenstellung verschiedener Produktkonfigurationen, je nach Domäne und Unternehmen, die das Werkzeug einsetzt.

4.2.1 Entwicklungsprozessdefinition

Da das Prozessmetamodell auf dem bereits existierenden Standard SPEM und deren Referenzimplementierung, der Unified Method Architecture (kurz UMA (Haumer, 2005)) basiert, standen zum Zeitpunkt dieser Arbeit zwei Implementierungen als Grundlage für den Prototyp zur Verfügung Zum einen das Open Source Werkzeug des Eclipse Process Framework Projekts und zum anderen das kommerzielle Werkzeug Rational

Method Composer[10] von IBM. Auf Grund der Verfügbarkeit wurde der EPF Composer als Werkzeug zur Prozessdefinition ausgewählt und erweitert. Die Erweiterung unterstützt die ersten beiden Schritte der beschriebenen Methodik in Abschnitt 3.3.

Method Content Definition

Die Bausteine eines Prozesses werden mit EPF in der *Authoring* Perspektive definiert. Hierzu zählen *Tasks*, *Work Products*, *Roles* und *Guidance*. Die *Authoring* Perspektive, dargestellt in Abbildung 40, unterteilt sich in drei Hauptbestandteile. Die *Library View* visualisiert alle Bausteine und die daraus erstellten Prozesse. In der *Editor Area* bearbeitet der Benutzer ausgewählte, bzw. neu erzeugte Elemente der *Library View*. Der dritte Bereich der Perspektive *Element Properties* erlaubt die Bearbeitung einzelner Schritte eines Prozesses (dargestellt in *Editor Area*). Für eine detaillierte Beschreibung der EPF Basisfunktionalitäten zur Erstellung von Elementen sei auf Tutorials verwiesen.

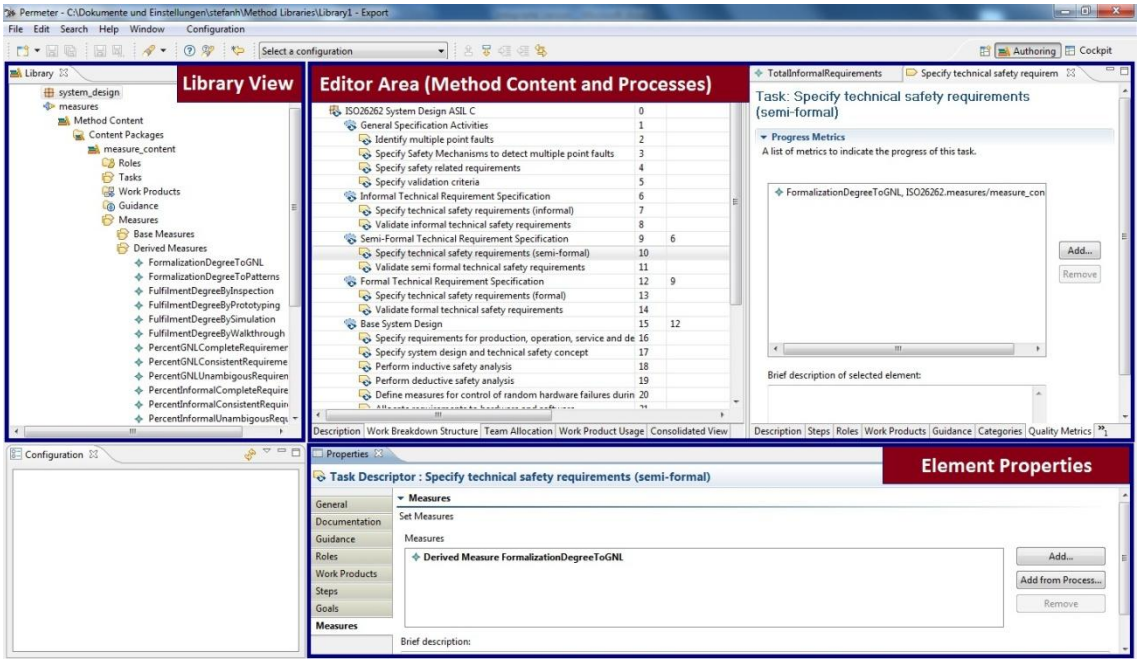

Abbildung 40: Übersicht Authoring Perspektive

Erweitert wurde der Editor zur Definition von Work Products mit der Möglichkeit einer formalen Definition (siehe Abbildung 41) mittels OCL zwecks konzeptioneller Integration von Prozess- und Produktsicht (siehe Abschnitt 3.2.3). Zur OCL Auswertung verwendet der Prototyp die Implementierung des OCL Standards für EMF basierte Modelle des Eclipse Model Development Tools[11] (kurz MDT) Projekts[12].

[10] http://www-01.ibm.com/software/awdtools/rmc/

[11] http://www.eclipse.org/modeling/mdt

[12] http://www.eclipse.org/modeling/mdt/?project=ocl#ocl

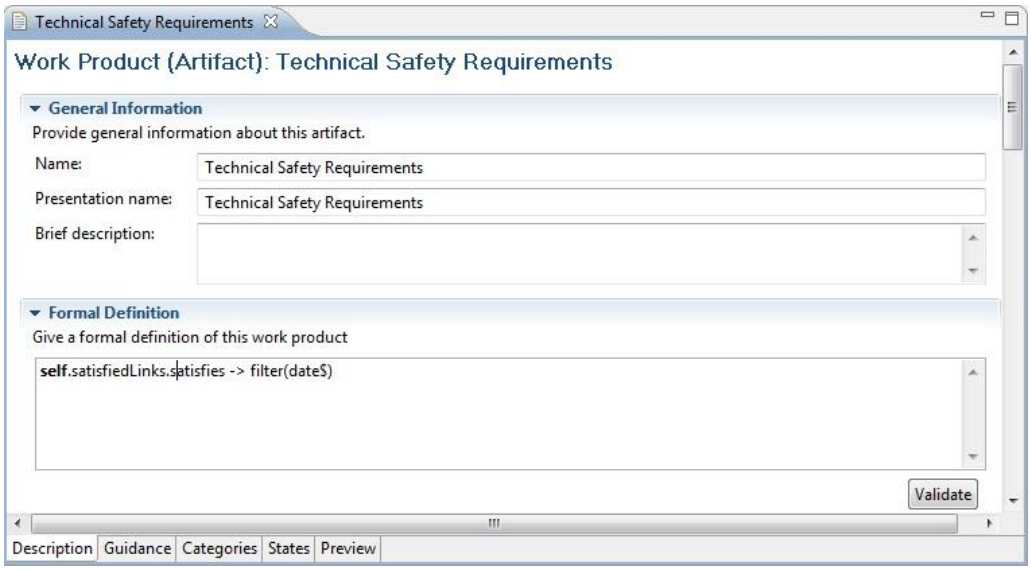

Abbildung 41: Editor - Definition Work Product „Technical Safety Requirements"

Jede Abfrage startet mit der Kontextvariable **self**, die sich üblicherweise auf eine einzelne bzw. eine Menge an Instanzen bezieht, gegen die eine Abfrage zu evaluieren ist. Die Kontexttyp (z.B. `SystemRequirement` oder `Component`) wird innerhalb des Werkzeugs automatisch wie folgt abgeleitet:

a) Das Work Product erweitert (`Extends`) ein anderes Work Product (z.B. „Formale Anforderungsspezifikation“ erweitert das Work Product „Anforderungsspezifikation“) → Der Typ der **self** Variable leitet sich aus dem Typ des Outputs der Definition des erweiterten Work Products ab.

b) Keine Erweiterung vorhanden → **self** ist vom Typ `Component`. Jede Abfrage geht grundsätzlich von einer Komponente aus. Abbildung 41 zeigt ein Beispiel für die Formalisierung von „Technical Safety Requirements“. **self** bezieht sich auf das Konzept `Component`, die Abfrage sucht zur Laufzeit alle Anforderungen, die mit einer `Component` Instanz über eine `Satisfy` Relation verbunden sind.

Eine Qualitätsmetrik ist nur automatisch gegen den aktuellen Entwicklungsstand auswertbar, wenn die Metrik sich auf ein mit OCL formalisiertes Work Product bezieht.

Prozessdefinition

Zusammen mit der Prozessdefinition wird die gültige Qualitätsdefinition (siehe 4.2.2) für diesen Prozess festgelegt. Der Prototyp integriert alle Funktionalitäten des Eclipse Process Frameworks für die Definition von Prozessen. Auch hier sei für eine detaillierte Beschreibung auf existierende Tutorials verwiesen.

4.2.2 Prozessorientierte Qualitätsdefinition

Entsprechend der konzeptionellen Erweiterung des SPEM Prozessmetamodells aus Abschnitt 3.2.2 erweitert die prototypische Umsetzung den EPF Composer um Funktiona-

litäten zur Definition von Qualitätsmetriken und –zielen. Die Erweiterungen fügen sich in die *Authoring* Perspektive ein und werden im Folgenden genauer beschrieben.

Qualitätsmetrik Definition

Qualitätsmetriken (im Werkzeug als *Measure* bezeichnet) sind wie Tasks und Work Products Bestandteil des *Method Contents*. Sobald diese Methodenbausteine definiert sind, folgt die Definition von Qualitätsmetriken.

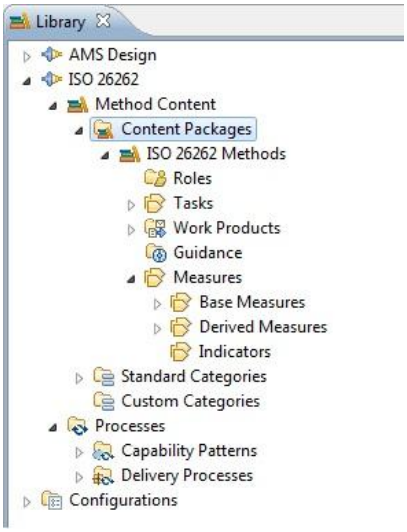

Abbildung 42: Library View

Die Erweiterung implementiert das Metamodell aus Abschnitt 3.2.2 und ermöglicht somit die Definition von *Base Measures, Derived Measures* und *Indicators* (siehe die Abbildung 42 dargestellte Library Ansicht). Die folgenden Reiter existieren in jedem Editor für eine neu angelegte Qualitätsmetrik:

- *Description*: Reiter zur Beschreibung von Metadaten über die Qualitätsmetrik, wie Name, Beschreibung, Version, Autor, Zweck, etc.
- *Definition*: Reiter für die formale Definition einer Qualitätsmetrik. Dieser Reiter ist Metrik spezifisch und wird im Laufe des Abschnitts für die verschiedenen Typen genauer beschrieben.
- *Work Products*: Reiter, um die ausgewählte Qualitätsmetrik mit einem oder mehreren Work Products zu verknüpfen, gegen die sie zu evaluieren sind. Qualitätsmetriken sind nur mit einem Task verknüpfbar, wenn sie auf mindestens ein Work Product definiert sind.

Um die definierten Qualitätsmetriken mit einem Task zu verknüpfen, wird im jeweiligen Editor (siehe Abbildung 43) im Reiter *Quality Metrics* Metriken für den Task ausgewählt.

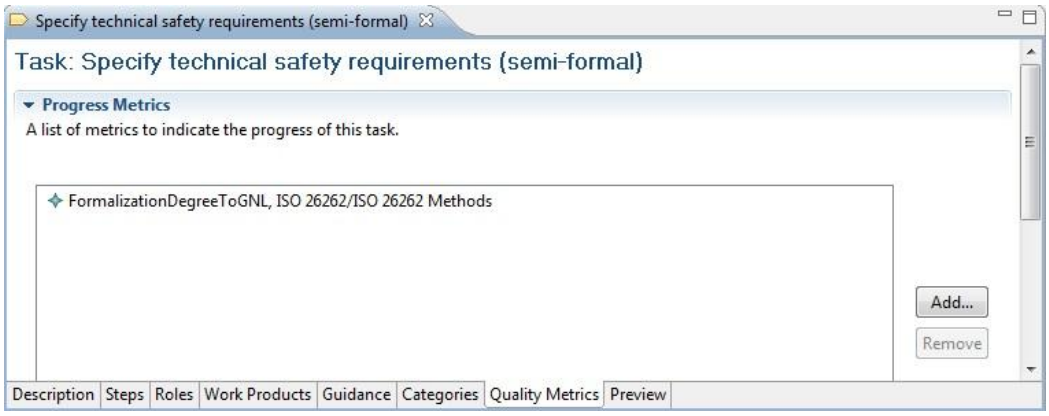

Abbildung 43: Definition von Qualitätsmetriken für Task „Specify technical safety requirements (semi-formal)“

Beispiel: Der Task „Specify technical safety requirements (semi-formal)“ ist vollständig durchgeführt sobald alle informellen Anforderungen über einen entsprechenden Link mit einer Anforderung vom Typ Guided Natural Language (kurz GNL) verknüpft sind. Der Formalisierungsgrad wird mit Hilfe der Qualitätsmetrik „FormalizationDegreeToGNL“ gemessen.

Unabhängig des Typs ist jede Qualitätsmetrik so zu definieren, dass sie genau einen Zahlenwert zurückliefert. Ansonsten werden Metriken zur Laufzeit als invalide interpretiert und nicht ausgewertet. Die korrekte Metrik Definition ist je nach Implementierung der verschiedenen Metrik Arten unterschiedlich sicherzustellen. Im Folgenden werden diese genauer beschrieben:

Base Measure Definition

Base Measures stellen die Anker zwischen einer Prozessdefinition und aktuellen Produktdaten dar und nutzen wie - bereits bei der Definition eines Work Products - OCL als Abfragesprache. Die folgenden in Abbildung 44 dargestellten Schritte werden durchgeführt, um eindeutig die durch das Produktmetamodell repräsentierten Instanzen zu identifizieren, die für die Auswertung eines Base Measures heranzuziehen sind:

1) Startpunkt der Auswertung ist der definierte Kontext eines instanziierten (z.B. Projektplan, siehe Erstellung in Abschnitt 4.2.3) Prozesses. Dies ist z.B. das gesamte System, eine Architektur oder eine Komponente.

2) Entsprechend des Work Products, auf dem der Base Measure definiert ist, wird eine Menge (Collection) von Elementen ausgewählt, gegen die der Base Measure evaluiert wird. Beispiel: Das Work Product „Requirements“ mit der Definition **self.satisfiedByLink.satisfies** liefert eine Menge an `SystemRequirement` Instanzen, die über eine `Satsify` Relation mit der Komponente aus 1) verknüpft sind.

3) Als letzter Schritt folgt der Base Measure spezifische Teil der Abfrage Beispiel: **self → size()** zählt die Anforderungen, die zu einem Zeitpunkt mit der

Komponente verbunden sind, auf den sich der betrachtete Prozess (Projektplan) bezieht.

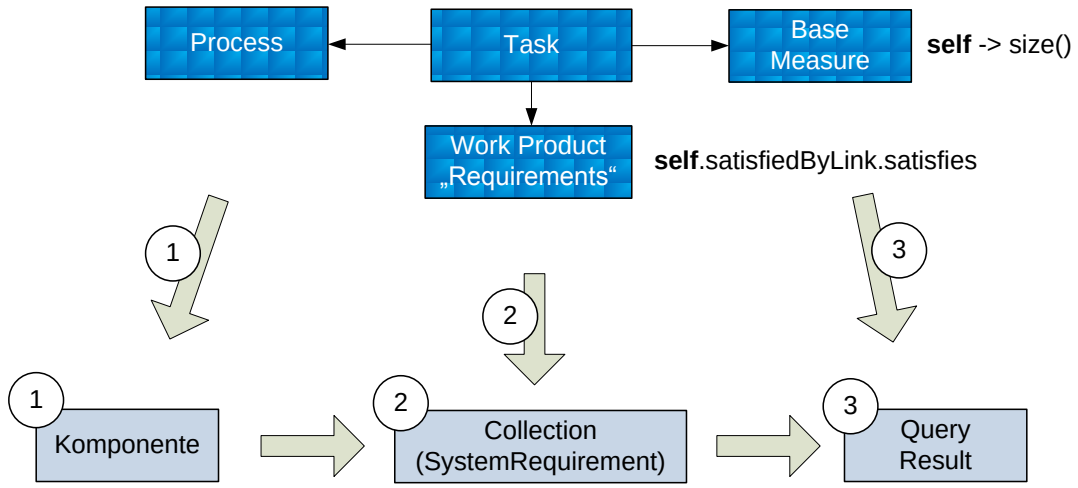

Abbildung 44: Ablauf zur Auswertung eines Base Measure

Neben den Standard Abfrageoperationen der OCL Spezifikation, stellt der Prototyp erweiterte Operationen zur Vereinfachung der Definition von Abfragen zur Verfügung:

- **filter(Date applicationDate) : List<VersionedElement> -** Filtert eine Menge an Elementen entsprechend des mit übergebenen Parameters „applicationDate“. Liefert eine Liste mit Elementen zurück, welche an dem gegebenen Datum gültig sind (d.h. die aktuellsten Versionen von Elementen deren Erstellungsdatum vor dem „applicationDate“ liegen).
- **requirementFulfilment(Date applicationDate, String category) : Double** – Berechnet den Erfüllungsgrad einer Anforderung zum Zeitpunkt des übergebenen Parameters “applicationDate”. Der Erfüllungsgrad errechnet sich basierend auf den mit der Anforderung verknüpften Analyseergebnissen. Optional besteht die Möglichkeit eine V&V Kategorie (z.B. Simulation oder Inspektion) mittels des Parameters „category“ anzugeben. Dies überprüft, ob der Nachweis der Anforderungserfüllung mittels einer bestimmten Methode durchgeführt wurde. Ist der Parameter null, wird der Erfüllungsgrad einer Anforderung unter Berücksichtigung aller vorhandenen Analyseergebnisse berechnet.
- **inLastDay(Date applicationDate) : Boolean** – Überprüft, ob ein Datum maximal einen Tag vor dem gegebenen “applicationDate” liegt.
- **inLastMonth(Date applicationDate) : Boolean** – Überprüft, ob ein Datum maximal einen Monat vor dem gegebenen “applicationDate” liegt.
- **inLastWeek(Date applicationDate) : Boolean** – Überprüft, ob ein Datum maximal eine Woche vor dem gegebenen “applicationDate” liegt.
- **lastLog(Date applicationDate) : VVLog** – Jede `VVProcedure` besitzt eine Menge an `VVLog` Objekten zur Speicherung der Ergebnisse der

`VVProcedure`. Diese Methode wird auf eine `VVProcedure` angewendet und liefert das aktuell gültige Verifikationsergebnis in Form eines `VVLog` zurück. Maßstab für die Auswahl ist ein übergebenes Datum.

Abbildung 45: Base Measure OCL Query Beispiel

Beispiel: Abbildung 45 zeigt den Reiter zur Definition einer Abfrage eines Base Measures, welcher auf einem Work Product „Technical Safety Requirements (informal)" definiert ist und die Anzahl informeller Anforderungen zählt. Die **self** Variable bezieht sich dementsprechend auf eine Menge an `SystemRequirement` Objekten, die mit Hilfe der size() Operation gezählt wird. Die Ausführung dieser Abfrage folgt auf die Abfrage des Work Products.

Derived Measure Definition

Ein Derived Measure führt eine mathematische Berechnung basierend auf den Ergebnissen anderer Qualitätsmetriken durch. Dementsprechend legt man bei der Definition eines Derived Measure im entsprechenden Reiter im Prototyp die folgenden zwei Aspekte fest:

- **Input Measures:** Die Qualitätsmetriken, deren Ergebnisse die Eingangsparameter für den Derived Measure sind. Nur Qualitätsmetriken, die auf Work Products mit demselben Typ definiert sind, sind hier auswählbar.
- **Mathematical Expression:** Eine textuelle Repräsentation einer mathematischen Funktion, wie die Ergebnisse der definierten Input Measures weiterzuverarbeiten sind.

Indicator Definition

Ein Indicator beschreibt komplexe Modelle zur Bewertung einer Qualitätsmetrik. Beispiele hierfür sind z.B. Bayes (Heckerman, 1996) oder Fuzzy Logic (Zadeh, 1965), (Berson, 1997) Klassifikatoren. Das Konzept des Indicator ist im Prototyp nicht implementiert. Denkbar wäre jedoch bereits existierende Data Mining Bibliotheken (z.B. Weka (Hall, 2009)) einzubinden und mit diesen Bibliotheken trainierte Modelle an dieser Stelle zu verwenden und wie jede andere Funktion basierend auf Eingangsparametern einen Ausgangswert zu berechnen.

Zieldefinition

Neben der in Abbildung 46 dargestellten Prozessdefinition erlaubt der Prototyp die Definition von prozessspezifischen Qualitätszielen, die von Prozessschritten oder Meilensteinen zu erfüllen sind.

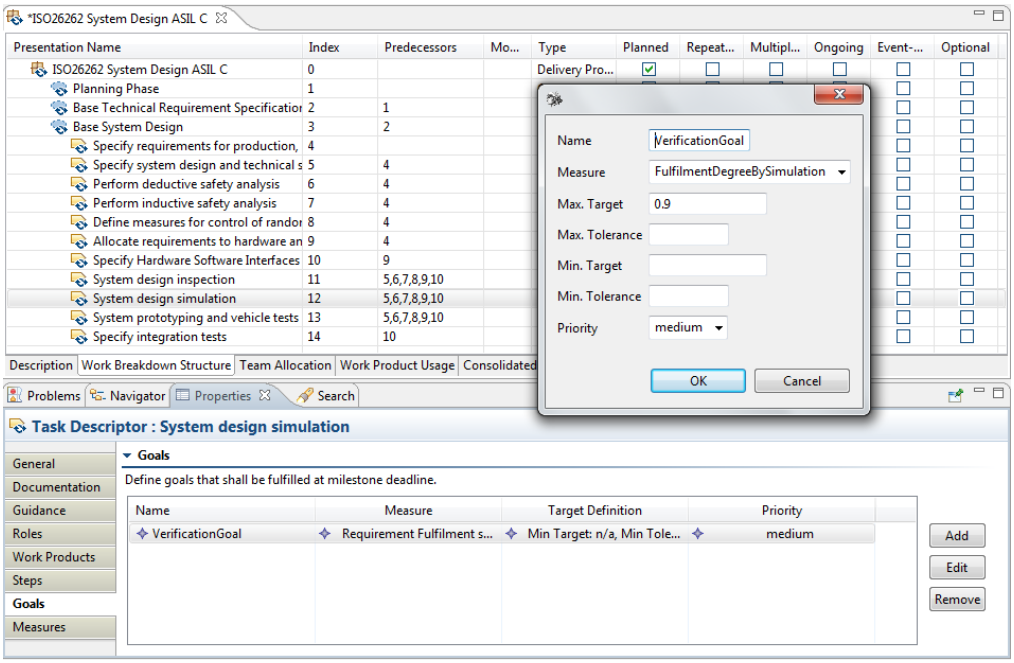

Abbildung 46: Definition von Qualitätszielen

Es besteht die Möglichkeit für jedes Prozesselement ein oder mehrere Qualitätsziele zu definieren. Hierzu ist das entsprechende Element im *Process Viewer* (siehe Abbildung 46) und der Reiter *Goals* in der unten erscheinenden Detailbeschreibung des Elements auszuwählen. Eine Tabelle zeigt alle bereits definierten Ziele. Beim Hinzufügen weiterer Ziele erlaubt ein Dialog die Auswahl einer Qualitätsmetrik und die Definition von Zielwerten und Toleranzen.

Zusammenfassung

Die bisher beschriebenen Teile des Prototyps unterstützen die ersten beiden Schritte des definierten Vorgehensmodells „Definiere Entwicklungsprozess“ und „Prozessorientierte Qualitätsdefinition“ (siehe Abschnitt 3.3.1). Entwicklungsprozesse wurden zusammen mit ihren Qualitätsmetriken und Qualitätszielen basierend auf der formalen Beschreibung des Produkts definiert und sind nun in Projekten anwendbar. Abbildung 47 demonstriert dies abschließend an einen Requirements Engineering Prozess, der auch im Anwendungsbeispiel für das ISO26262 basierte Management funktionaler Sicherheit Verwendung findet. Jeder Prozessschritt (siehe z.B. Abbildung 47 „Specify technical safety requirements (semi-formal)) besitzt automatisch Qualitätsmetriken, solange für ihre äquivalenten Tasks im *Method Content* ebenfalls Metriken definiert sind.

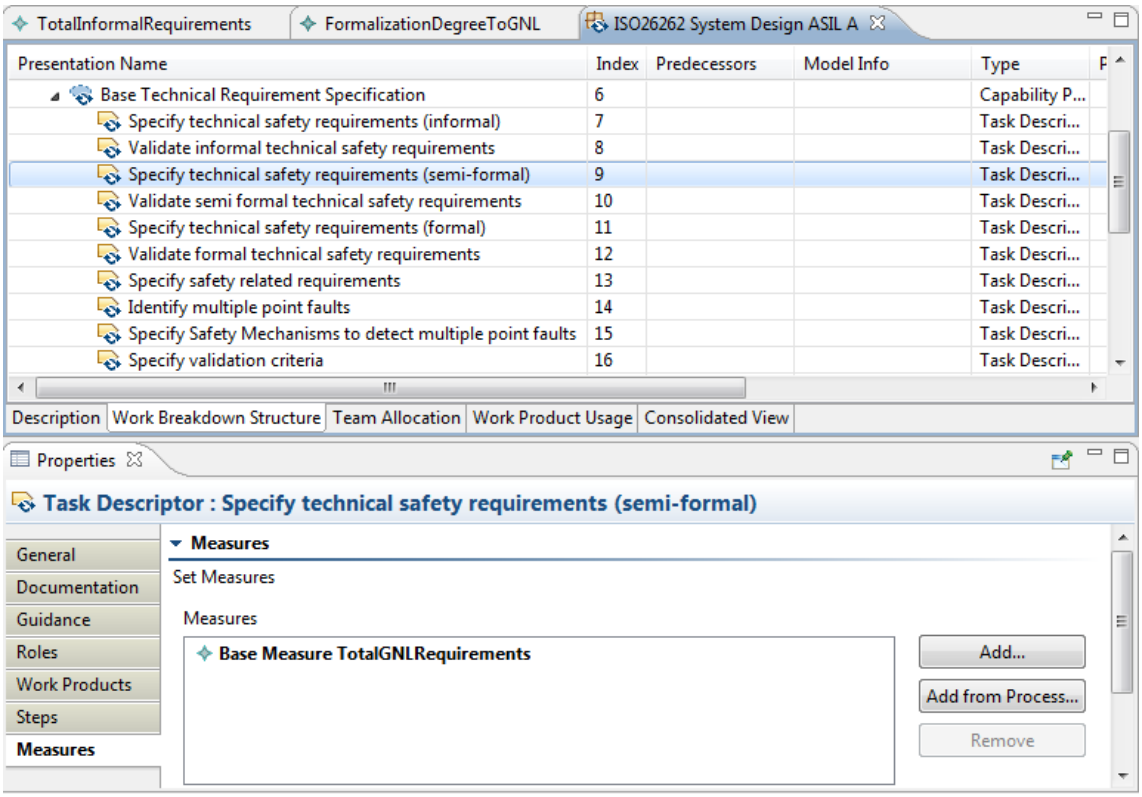

Abbildung 47: Prozessdefinition

Bei Bedarf ist es möglich die definierten Qualitätsmetriken auf den jeweiligen Prozess zuzuschneiden. Qualitätsmetriken aus dem *Method Content* sind deaktivierbar oder können nachträglich hinzugefügt werden.

4.2.3 Projektplanerstellung

Um eine prozessorientierte Qualitätsdefinition in einem Projekt anzuwenden, implementiert der Prototyp Funktionalitäten zur Kalibrierung von Prozessen und der zugehörigen Qualitätsdefinition. Ein Wizard erlaubt die Erstellung von Projektplantemplates für ein oder mehrere Teile des zu entwickelnden Systems. Darüber hinaus enthält der Prototyp eine Anbindung an das Projektmanagement Werkzeug MS Project, um die Planung von Deadlines und Ressourcen durch ein bereits existierendes Projektmanagementwerkzeuge zu unterstützten. Existieren Adapter für andere Projektmanagementwerkzeuge, sind auch diese als Planungswerkzeug verwendbar. Sowohl Projektplanerstellung als auch die MS Project Anbindung werden im Folgenden kurz beschrieben.

Erzeugung von Projektplan Templates

Zur Erzeugung eines Projektplan Templates dient die *Model Integration* Perspektive, welche eine Übersicht aller Projekte und der dort vorhandenen Modelle liefert. Ein Projekt enthält zwei Ordner: einen für Projektpläne und einen für Entwicklungsdaten, repräsentiert durch Instanzen des Produktmetamodells. Projektpläne werden für Design Modelle oder Teile dieser instanziiert. Ein hierfür entwickelter Wizard ordnet eine oder mehrere Prozessdefinitionen Komponenten oder ganzen Design Modellen zu. Abbil-

dung 48 stellt diesen Wizard dar, in dem für eine Komponente der System Architektur die Definition eines Sicherheitsprozesses nach ISO26262 ausgewählt wurde.

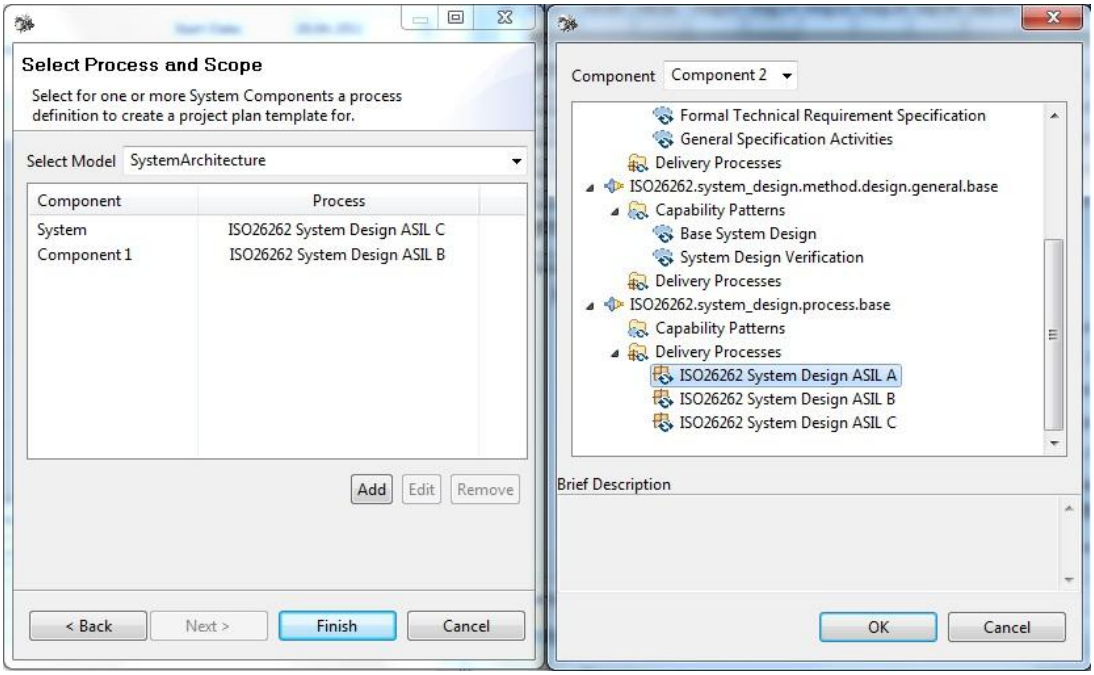

Abbildung 48: Projektplanerzeugung

Ergebnis ist ein initialer Projektplan, der einen kalibrierten Entwicklungsprozess inklusive einer angepassten Qualitätsdefinition für ein konkretes Produkt repräsentiert.

Anbindung Projektmanagementwerkzeug MS Project

Um die Funktionalitäten aktueller Projektmanagementwerkzeuge zu unterstützen, enthält der Prototyp einen Adapter für die Anbindung von MS Project. Mit Hilfe des Eclipse Modelling Frameworks und der XSD Schema Definition von MS Project 2010[13] wurde eine API zum Erstellen und Laden von MS Project konformen XML Dateien generiert.

Der Adapter erlaubt die drei in Abbildung 49 dargestellten Operationen.

1. Initialer Export

Liegt noch kein Projektplan in der MS Project XML Repräsentation vor, wird dieser erzeugt. Das generierte Projektplan Template im Projektmetamodell Format wird durch die Transformation in das MS Project XML Format überführt und ist fortan in MS Project bearbeitbar.

[13] http://msdn.microsoft.com/en-us/library/bb428843.aspx

2. Synchronisiere Plandaten

Der Adapter erlaubt die Synchronisation zwischen MS Project und dem Projektmetamodell Format, um so annotierte Start und Endtermine aus der MS Project Datei zu übernehmen.

3. Regelmäßiger Export von Fortschrittsdaten

Der Adapter exportiert die auf Qualitätsmetriken basierten Fortschrittsinformationen nach MS Project, um diese so auch dort verfügbar zu machen.

Abbildung 49: Datenaustausch zwischen Prototyp und MS Project

Analoge Datentransformationen sind auch für andere Projektmanagement Werkzeuge umsetzbar.

4.2.4 Qualitätsbewertung

Die *Cockpit* Perspektive (dargestellt in Abbildung 50) stellt die Ergebnisse der Qualitätsbewertung in verschiedenen Ansichten dar.

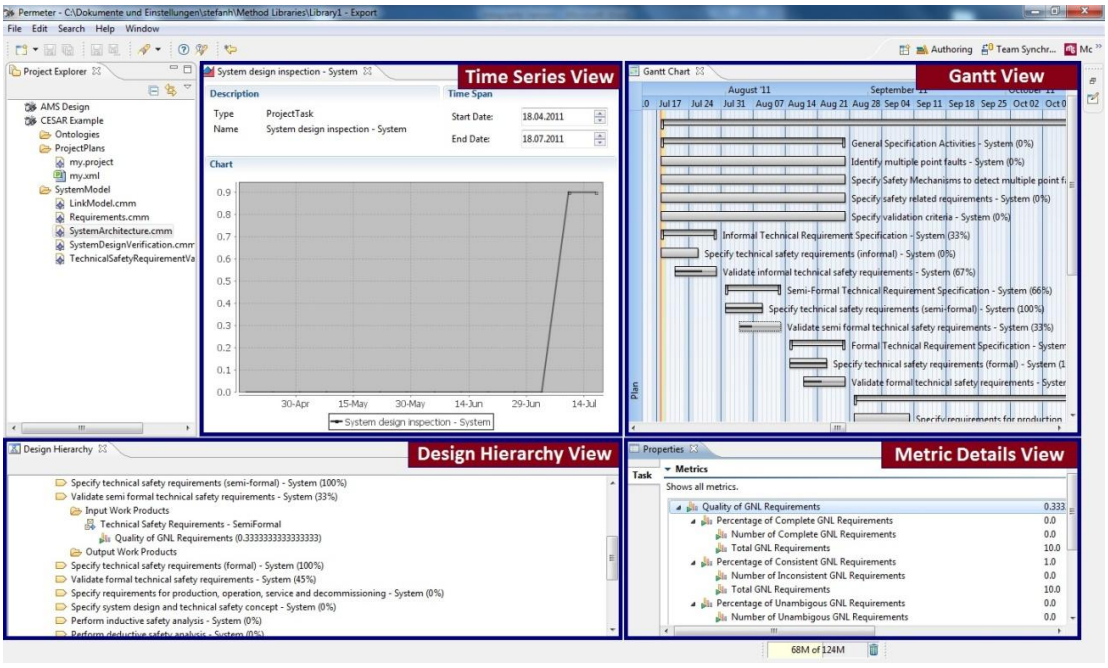

Abbildung 50: Übersicht Cockpit Perspektive

Im weiteren Verlauf dieses Abschnitts wird Funktion und Darstellung jeder Ansicht erläutert.

Gantt View

Die *Gantt View* visualisiert alle Projektpläne. Ein **Gantt-Diagramm** ist ein nach dem Unternehmensberater Henry L. Gantt (1861–1919) benanntes Instrument, das die zeitliche Abfolge von Tasks grafisch in Form von Balken auf einer Zeitachse darstellt. Das Gantt Diagramm ordnet Tasks entsprechend seiner Laufzeit an der Zeitachse an (siehe Abbildung 50 rechts oben). Die zugeordneten Qualitätsmetriken und –ziele sind die Grundlage für die Fortschrittsbewertung eines Tasks. Die *Metric Details View* (siehe Abbildung 50 rechts unten) visualisiert die hierarchische Definition und deren aktuellen Werte jeder Qualitätsmetrik für einen ausgewählten Task in der *Gantt View*.

Design Hierarchy View

Die *Design Hierarchy View* stellt die Auswertungsergebnisse entlang des hierarchischen Aufbaus eines Design Modells dar, berücksichtigt aber auch den Bezug zum Entwicklungsprozess. Ausgehend von einer Komponente lässt sich wie folgt in die Qualitätsbewertung eintauchen:

1) Tasks: Jeder Komponente (in Abbildung 51 „System“) sind entsprechend des für die Komponente instanziierten Prozesses Tasks zugeordnet. Jeder Task besitzt eine prozentuale Angabe über seinen aktuellen Fortschritt. In Abbildung 51 z.B. 90% Fortschritt für „Specify technical safety requirements (semi-formal)“.
2) Work Products: Jeder Task hat In- und Outputs (z.B. „Technical Safety Requirements (informal)“ als Input des eben genannten Tasks). Qualitätsmetriken sind den Work Products zugeordnet.
3) Qualitätsmetriken: Für jede Qualitätsmetrik (z.B. „Formalization Degree from Informal to GNL“ in Abbildung 51) ist das aktuelle Ergebnis dargestellt. Jede Metrik lässt sich bis zu ihren Base Measures aufklappen.

Aus der Erfüllung zugeordneter Tasks lässt sich eine Qualitätssausage über die Komponente ableiten. Bei dieser Aggregation handelt es sich stets um eine kontextspezifische Qualitätsbewertung der Komponente aus Prozesssicht. Verschiedene Prozesse führen zu unterschiedlichen Ergebnissen. Ebenfalls wichtig ist die Gewichtung der einzelnen Tasks, z.B. durch die Berechnung eines relativen Gewichtungsfaktors, der das Verhältnis zwischen dem Task zugeordneten Aufwänden und dem Gesamtaufwand darstellt.

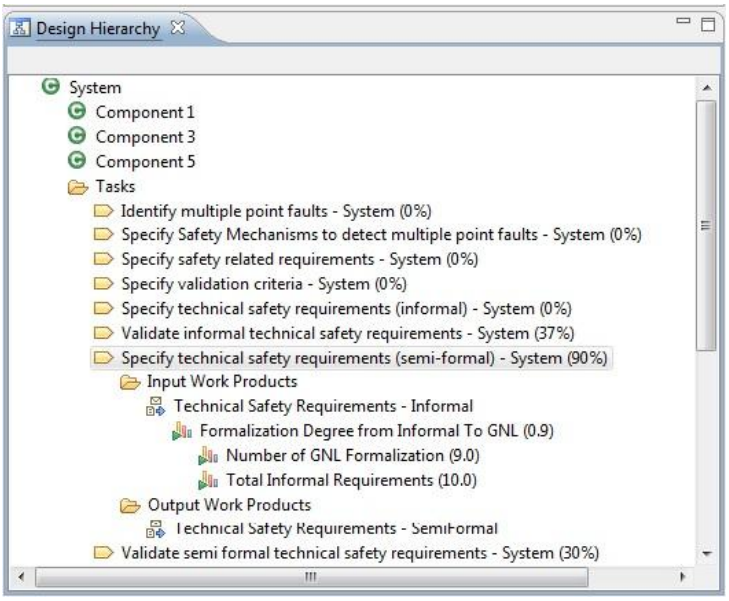

Abbildung 51: Design Hierarchy View

Ausgehend von einem Element in der *Design Hierarchy View* lässt sich die Entwicklung über die Zeit einer Metrik oder eines ganzen Tasks darstellen.

Time Series View

Für die folgenden Elemente erlaubt der Prototyp die Darstellung der Wertentwicklung über die Zeit mittels implementierter Charts (Beispiel siehe Abbildung 50 links oben):

- Task Fortschritt: Stellt die prozentuale Fortschrittsentwicklung über die Zeit eines Tasks bzw. seiner definierten Qualitätsziele dar.
- Meilensteinerfüllungsgrad: Stellt die Entwicklung des Meilensteinerfüllungsgrads über die Zeit dar.
- Qualitätsmetrik: Stellt die Entwicklung der Metrik über die Zeit dar.

Um die Bewertung auf Basis aktueller Entwicklungsdaten durchzuführen, verwendet der Prototyp existierende Technologien für die Datenerfassung und –speicherung. Die entsprechende Umsetzung dieser Komponente wird daher nur kurz skizziert:

Für die Datenerfassung wird grundsätzlich auf das Konzept der Modelltransformationen zurückgegriffen, wie es z.B. auch von Hausmann (Hausmann, 2008) vorgeschlagen wurde. Eine Modelltransformation implementiert eine Abbildung zwischen einem proprietärem Datenformat und dem Produktmetamodell. Erweiterungsmöglichkeiten (Extension Points) erlauben die Einbindung beliebiger Modelltransformationen, solange sie ein Modell liefern, welches konform zum Produktmetamodell ist.

Die Datenspeicherung erfolgt dateibasiert innerhalb eines Workspace unter Verwendung des EMF Frameworks, welches alle Funktionalitäten zur Erzeugung und Manipulation von Modellen zur Verfügung stellt. Für die Konsistenz der Verknüpfungen zwischen den einzelnen Teilmodellen sorgt ebenfalls das EMF Framework. Konzepte zur

Versionierung von Elementen innerhalb der implementierten EMF Metamodelle erlauben die Erfassung der Entwicklung über die Zeit.

4.3 Zusammenfassung

Der entwickelte Prototyp implementiert verschiedene Komponenten zur Umsetzung des in Kapitel 3 aufgestellten Konzepts und adressiert damit insbesondere die Schwachstelle der fehlenden Werkzeugunterstützung (siehe **Anforderungen 11 bis 13**). Insbesondere ist der Prototyp für verschiedene Anwendungsbereiche kalibrier- bzw. erweiterbar, speziell durch die Einbindung von Transformationen für beliebige Datenquellen. Der entwickelte Prototyp erfüllt damit alle Voraussetzungen, um als Grundlage für die Durchführung der Evaluation zu dienen.

5 Evaluation

Die beiden vorherigen Kapitel zeigten bereits, dass Konzept und Implementierung die zu Beginn von Kapitel 3 aufgestellten (technischen) Anforderungen abdecken. Auf Grund der Komplexität der entwickelten Lösung besteht die Notwendigkeit einer ausführlichen Evaluation der Ergebnisse. Zu diesem Zweck führt dieses Kapitel auf Basis des entwickelten Prototyps entsprechende Tests und Untersuchungen durch. Übergreifende Ziele der Evaluation sind wie folgt:

1) **Überprüfung der Zielerfüllung:** Primär untersucht die Evaluation die Erfüllung der in Abschnitt 1.4 aufgestellten Ziele dieser Arbeit, welche sich sowohl aus existierender Literatur als auch aus Industrieanforderungen des Projekts CESAR ableiten ließen. Die Ziele stellen Motivation und Grundlage der Konzeption. Ihre Erfüllung ist für den Erfolg dieser Arbeit maßgebend. Methodisch untersucht die Evaluation dies mit Hilfe konkreter Fragestellungen aus zwei Anwendungsbeispielen.

2) **Demonstration der Anwendbarkeit:** Neben der Untersuchung der Zielerfüllung illustriert die Evaluation dem Leser die Anwendbarkeit der entwickelten Lösung im Rahmen industrieller Entwicklungsprozesse.

3) **Identifikation möglicher Schwachstellen und Optimierungspotentiale:** Keine Arbeit liefert eine Antwort auf alle Eventualitäten und Anwendungskontexte, noch ist sie für alle gleich gut geeignet. Die Evaluation liefert Erfahrungen mit dem Ansatz, aus denen sich Entscheidungshilfen für oder gegen den Einsatz der Methodik ableiten lassen. Wann eignet sich der Ansatz z.B. besonders gut, wann weniger gut? Die Evaluation versucht eine Antwort auf diese Frage zu liefern.

Die Evaluation beschäftigt sich mit Fallbeispielen aktuell hochgradig relevanter Bereiche der Automotive Elektronikentwicklung. Ein Anwendungsbeispiel wurde in Zusammenarbeit mit der Robert Bosch GmbH durchgeführt und hat einen industriellen Analog Mixed Signal (kurz AMS) Entwicklungsprozess (Abschnitt 5.1) als Betrachtungsgegenstand. Die Durchführung erfolgte innerhalb des Forschungsprojekts PRODUKTIV+ (Alt, 2007). Das Beispiel eignet sich insbesondere aus den folgenden zwei Gründen: Zum einen umfasst das Anwendungsbeispiel eine entwicklungsbegleitende Qualitätsbewertung und deren Anbindung an Projektkontrolle und –steuerung. Somit eignet es sich insbesondere zur Evaluierung des Modellierungsframeworks, da es sowohl die Produkt- als auch die Prozesssicht berücksichtigt und die meisten Ziele dieser Arbeit abdeckt. Zum anderen erfolgte die Evaluation im Rahmen eines industriellen Entwick-

lungsprozesses, was dem Anspruch der Anwendungsorientierung dieser Arbeit zu Gute kommt.

Abschnitt 5.2 beschreibt die Anwendung der entwickelten Methodik für das Management der funktionalen Sicherheit gemäß des Sicherheitsstandards ISO26262, ein Standard für die Entwicklung sicherheitskritischer eingebetteter Systeme im Bereich Automotive. Der Standard betrachtet sowohl die Entwicklung des gesamten Elektroniksystems eines Fahrzeugs, als auch Hard- und Softwareentwicklung. Das Anwendungsbeispiel konzentriert sich auf Ausschnitte der System Design Phase. Ausgewählt wurde dieses Anwendungsbeispiel aus den folgenden zwei Gründen: Zum einem beinhaltet der ISO26262 Referenzprozess verschiedene variable Abschnitte, um die Kalibrierung einer unternehmensweiten prozessorientierten Qualitätsdefinition auf ein spezielles Projekt zu erproben. Es eignet sich daher insbesondere zur Evaluierung von **Ziel 2 - Effiziente Kalibrierung auf einzelne Projekte**. Zum anderen handelt es sich um einen standardisierten Prozess, der von allen Unternehmen im Bereiche der sicherheitskritischen Automobilelektronik zu berücksichtigen ist. Der betrachtete Prozess ist damit hinreichend repräsentativ, um bei positiver Evaluierung von einer grundsätzlichen Anwendbarkeit der Methodik bei verschiedenen Unternehmen zu sprechen. Durchgeführt wurde dieses Anwendungsbeispiel im Rahmen des industrienahen Forschungsprojekts CESAR. Abschnitt 5.3 gibt schließlich eine Zusammenfassung und eine kritische Reflektion der Arbeit und seiner Evaluierungsergebnisse mit den in Kapitel 1 aufgestellten Zielen.

5.1 Anwendungsbeispiel Analog Mixed Signal ASIC Design

Das folgende Anwendungsbeispiel beschreibt die Evaluation der entwickelten Methodik und des zugehörigen Frameworks in einem industriellen Umfeld bei der Robert Bosch GmbH. Innerhalb des Beispiels half die Methodik nicht nur dabei, den aktuellen Qualitätsstand in einem Analog Mixed Signal (kurz AMS) Entwurfsprozess zu überwachen, die Ergebnisse wurden ebenfalls für die Optimierung von Projektabläufen verwendet. Teile der Ergebnisse wurden in (Häusler, 2009) und (Häusler, 2010) veröffentlicht.

5.1.1 Anwendungskontext

AMS Designs enthalten sowohl analoge als auch digitale Funktionen auf einem Chip. Während jedoch die Entwicklung der digitalen Schaltungen heute im Wesentlichen sprachbasiert, hoch automatisiert und unter Verwendung von Synthesetools erfolgt, ist die Entwicklung von Analogblöcken zu einem großen Teil „manuell“, sehr interaktiv und applikationsabhängig. Dies macht aktuelle AMS Entwicklungsprozesse sehr aufwändig. Dies wäre vernachlässigbar, wenn es sich dabei nur um einen eher kleinen Anteil an der Gesamtentwicklung handeln würde. Die Welt spricht jedoch analog, was in Zeiten immer stärker vernetzter Systeme mit ihrer Umwelt zu komplexeren Produkten und damit aufwändigeren Entwicklungsprozessen führt. Die Komplexität der Entwicklung nicht digitaler Komponenten wie Sensoren und Aktuatoren oder auch der Stromversorgung in Kombination mit Digitalen Komponenten sind zukünftige Herausforde-

rungen der Halbleiterentwicklung (vgl. (Arden, 2010)). In diesem Umfeld schafft das entwickelte Framework Transparenz für ein auch in Zukunft unbedingt zu beherrschenden Entwurfsbereich.

Demonstriert wurde dies in Zusammenarbeit mit der Robert Bosch GmbH, einem Automobilzulieferer für elektronische Kraftfahrzeugkomponenten. Im Projekt PRODUKTIV+ entwickelte die beteiligte Abteilung eine Lösung zur entwicklungsbegleitenden Prozessüberwachung und –steuerung, um Produktqualität bereits während der AMS-Entwicklung zu bestimmen und bei Problemen möglichst frühzeitig Gegenmaßnahmen durchzuführen. Ergebnis war der in Abbildung 52 dargestellte Regelkreis (Häusler, 2010).

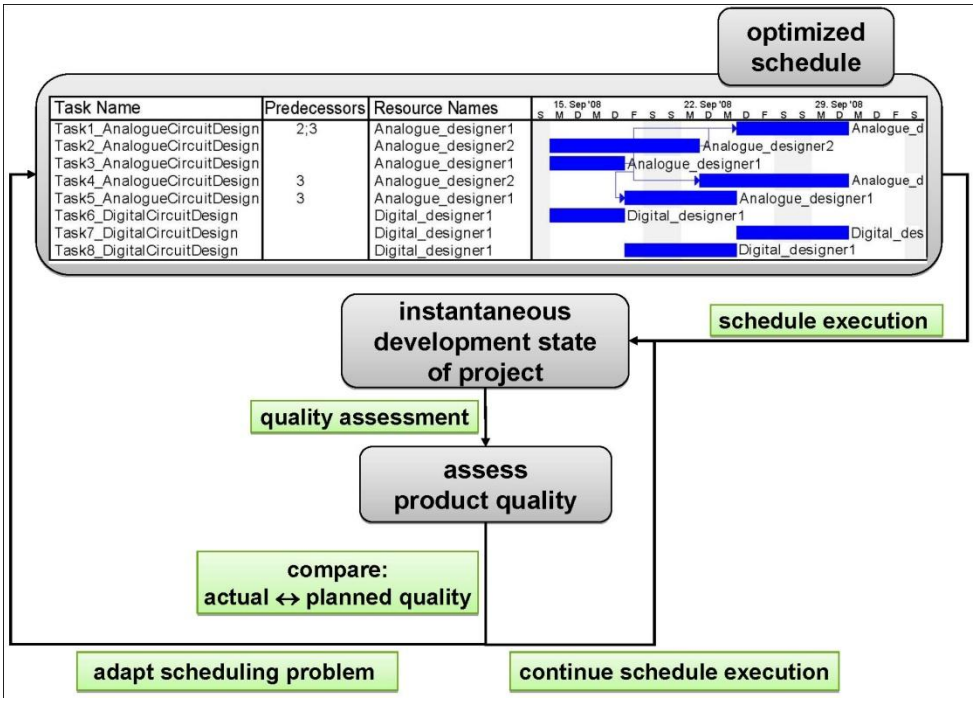

Abbildung 52: Gesamtarchitektur Bosch Anwendungsbeispiel

Im Zentrum steht das in dieser Arbeit entwickelte prozessorientierte Qualitätsmonitoring. Regelmäßig durchgeführte Qualitätsbewertungen werden mit Soll-Daten der Projektplanung abgeglichen und dienen als Eingang für entwickelte Algorithmen zur Projektplanoptimierung (Blaschke, 2009a), (Blaschke, 2009b), (Blaschke, 2010).

5.1.2 Ziele

Die Evaluation untersuchte Zielerfüllung am Gegenstand eines Analog-Mixed Signal Design Prozesses. Der folgende Abschnitt listet die durch die Evaluation untersuchten konkreten Fragestellungen auf und verknüpft diese mit jeweils einem Ziel dieser Arbeit (siehe Abschnitt 1.4). Eine positive Antwort auf die Fragen führt zu einer positiven Antwort bzgl. des Nachweises der Zielerfüllung:

1) Eignet sich der entwickelte Modellierungsansatz zur Abbildung der durch Ingenieure entwickelten Qualitätsmetriken zur Überwachung des aktuellen Entwicklungsstands? → **Ziel 1: Einheitliche Qualitätsdefinition.**
2) Eignet sich das entwickelte Produktmetamodell für die Abbildung aller anfallender V&V Ergebnisse, um die aufgestellte Qualitätsdefinition mit Ist-Daten abzugleichen? → **Ziel 3: Integration V&V Ergebnisse.**
3) Ermöglicht der Ansatz eine regelmäßige entwicklungsbegleitende Auswertung und die Analyse von Trends? → **Ziel 5: Entwicklungsbegleitende Auswertung.**
4) Ermöglicht das Qualitätsmodellierungsframework die Auswertung von Projekt- und Meilensteinzielen? Identifiziert es Abweichungen, um Projektsteuerungsmaßnahmen durch eine Projektplanungsoptimierungskomponente zu initiieren? → **Ziel 6: Berücksichtigung von Prozess- und Projektkontext.**
5) Erlaubt das entwickelte Qualitätsmodellierungsframework eine automatische Datenerfassung aller notwendigen Produktdaten bis hin zu einer automatisierten Aggregation und Interpretation entsprechend definierter prozessorientierter Qualitätsmetriken? → **Ziel 7: Automatische Datenerfassung und –integration.**
6) Ist ein Qualitätsmonitoring mittels des Qualitätsmodellierungsframeworks möglich, ohne dabei die existierenden Entwicklungsprozesse zu beeinflussen? → **Ziel 8: Anpassbarkeit an bestehende Entwicklungsprozesse.**

Im Zentrum der Evaluation steht in diesem Anwendungsbeispiel daher zum einen die Ausdrucksmächtigkeit des Produktmetamodells alle qualitätsrelevanten Daten abzubilden, die Ausdrucksmächtigkeit alle gewünschten Qualitätsmetriken abzubilden, sowie die Prozess- und Produktmetamodell Integration, um eine Prozess- und Projektkontext abhängige Bewertung durchzuführen.

5.1.3 Durchführung

Die Durchführung des Anwendungsbeispiels folgt dem in Abschnitt 3.3 beschriebenen Vorgehensmodell und wird entlang der dort definierten Schritte im Folgenden beschrieben.

Prozessdefinition

Die folgende Sektion beschreibt den im Fallbeispiel betrachteten Ausschnitt eines AMS Entwicklungsprozesses (Übersicht siehe Abbildung 53). Der Prozess erhebt dabei keinen Anspruch auf Vollständigkeit und legt mit den dargestellten Phasen Fokus auf die Verifikation in den einzelnen Phasen des Entwicklungsprozesses. Grau dargestellte Verifikationsabschnitte sind zur Vollständigkeit enthalten, wurden im Anwendungsbeispiel jedoch nicht weiter betrachtet.

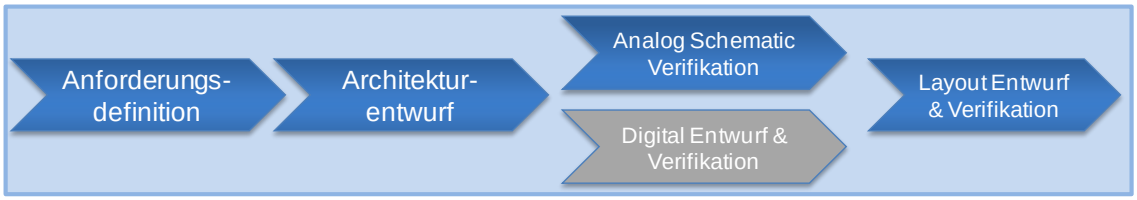

Abbildung 53: Fokussierte Entwicklungsphasen des AMS Entwicklungsprozesses

Die folgende Tabelle beschreibt für jeden der betrachteten Entwicklungsschritte dessen In- und Outputs:

Task	Input	Output
Anforderungsdefinition	Kundenanforderungen	Spezifikation
Architekturentwurf	Spezifikation	Systemarchitektur
Analog Schematic Verifikation	Schematic, Testspezifikation	Verifikationsergebnisse Analog Tests
Layout Entwurf & Verifikation	Spezifikation, Schematic	Layout Struktur, Layout Verifikationsergebnisse

Tabelle 12: Tasks mit In- und Output

Mit Hilfe der prototypischen Umsetzung wurden die einzelnen Tasks und Work Products beschrieben und zwei Hauptprozessbausteine definiert. Zum einen die Anforderungsdefinition und Architekturentwicklung für die Gesamtschaltung, zum anderen die Verifikation der Analogblöcke auf Schematic Ebene sowie Layout Entwurf und dessen Verifikation. Diese Bausteine sind im weiteren Verlauf Grundlage der schaltungsspezifischen Prozessinstanziierung. Der Entwicklungsprozess mit seinen einzelnen Schritten und In- und Outputs ist die Grundlage für die nun folgende Qualitätsdefinition.

Prozessorientierte Qualitätsdefinition

Für drei Prozessschritte wurden Qualitätsmetriken entwickelt. Dies umfasst die Bereiche Analog/Digital Layout, Analog Verifikation, sowie die Verifikation von Flächenanforderungen (siehe Abbildung 54).

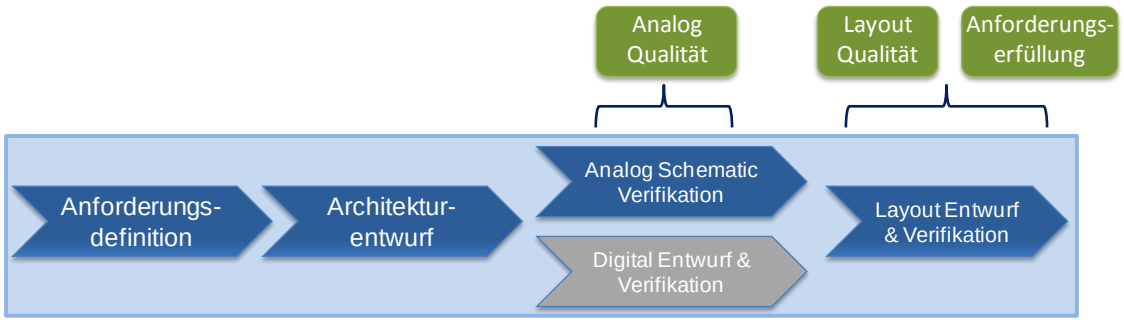

Abbildung 54: Qualitätsfortschrittsmetriken für einzelne Phasen

Die drei Qualitätsmetriken wurden entsprechend der folgenden Definition modelliert.

***Derived Measure*: Analog Qualität**

Work Product: Schematic; ***Task:*** Analog Schematic Verifikation

Beschreibung: Auf Grund des signifikanten Aufwands für analog Komponenten in AMS Schaltungen ist die Bewertung ihres Verifikationsgrads ein wichtiger Bestandteil der Qualitätsbewertung. Dieser Abschnitt beschreibt den hierarchischen Aufbau der Qualitätsmetrik zur Berechnung der Analog Frontend Designqualität, welche als der Fortschrittsindikator für den Schritt „Analog Schematic Verifikation“ der Analog Blöcke des AMS Designs dient. Das abschließende Kapitel dieses Fallbeispiels betrachtet die Qualitätsmetrik kritisch und schlägt Weiterentwicklungsmöglichkeiten vor. Den hierarchischen Aufbau der Qualitätsmetrik zeigt Abbildung 55.

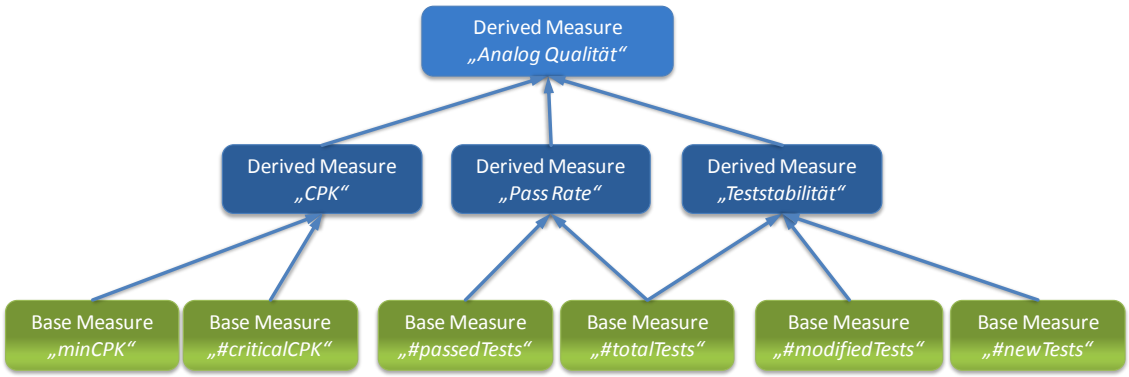

Abbildung 55: Analog Frontend Qualitätsmetrik

Der Designqualitätswert berechnet sich aus dem gewichteten arithmetischen Mittel der drei Derived Measures „Pass Rate“, „CPK“ und „Teststabilität“ der Analogzellen Tests. Im Fallbeispiel wurden diese Qualitätsmetriken zunächst gleich gewichtet.

$$\frac{k_1\,\mathrm{PassRate} + k_2\,\mathrm{CPK} + k_3\mathrm{Teststabilität}}{\sum_{j=1}^{3} k_j}$$

Es folgt eine Erläuterung der drei enthaltenen Metriken

***Derived Measure:* CPK**

Work Product: Schematic; ***Task:*** Analog Schematic Verifikation

Beschreibung: Der CPK ist ein Prozessfähigkeitsindex zur statistischen Bewertung eines Prozesses in der Produktionstechnik. Dieser Index wurde auf die Entwicklung übertragen und bewertet das Risiko, die Spezifikationsziele der Analogzellen Tests auch nach der Fertigung noch einhalten zu können. Die Faustregel lautet: Je näher Testergebnisse den Spezifikationsgrenzen kommen, desto höher ist das Risiko, diese nach der Fertigung auf Grund von Prozessvarianz doch noch zu verfehlen. In den CPK fließt zum einen der minimale CPK aller betroffenen Messwerte ein. Zum anderen wird die Anzahl der Messwerte gezählt, die unter einem kritischen CPK Wert liegen.

Die Berechnung des CPK's basiert auf den Angaben aus Spezifikationsziel und aktuell gemessenen Werten. Der Abstand zur maximalen und minimalen Grenze berechnet sich mit Hilfe der folgenden beiden Formeln:

$$maxCPK = \frac{limitHigh - 0.5\ (valueHigh + valueLow)}{0.5\ (valueHigh - valueLow)}$$

$$minCPK = \frac{0.5\ (valueHigh + valueLow) - \ limitLow}{0.5\ (valueHigh - valueLow)}$$

Der kleinere der beiden Werte wird als minimaler CPK Wert gesetzt. Ein Wert von 1 bedeutet, dass der gemessene Wert genau auf einer der beiden Spezifikationsgrenzen liegt, mit steigendem CPK wird der Abstand zur Grenze größer. In der aktuellen Version des Qualitätsmodells nimmt die „CPK" Metrik je nach Belegung der Werte „minCPK" und „#criticalCPK" die Werte 0, 0.5 oder 1 an. Die Klassifizierung erfolgt unter Verwendung der folgenden Funktion:

$$CPK = 1, wenn\ minCPK > 1.01$$
$$CPK = 0.5\ wenn\ 1.005 < minCPK \leq 1.01\ und\ \#criticalCPK < 5$$
$$CPK = 0, wenn\ minCPK \leq 1.005\ oder\ \#criticalCPK \geq 5$$

Eine optimale Einteilung der Grenzwerte war nicht primärer Gegenstand der Evaluierung, weshalb über die Eignung der gewählten Werte an dieser Stelle keine Aussage zu treffen ist. Je nach Bedarf ist diese Metrik Definition anpassbar.

Derived Measure: **Pass Rate**

Work Product: Schematic; ***Task:*** Analog Schematic Verifikation

Beschreibung: Die „Pass Rate" Qualitätsmetrik gibt den prozentualen Anteil erfolgreicher Tests von Analogzellen an und berechnet sich nach folgender Formel:

$$PassRate = \frac{\#passedTests}{\#totalTests}$$

Derived Measure: **Teststabilität**

Work Product*:* Schematic; ***Task:*** Analog Schematic Verifikation

Beschreibung: Die dritte Qualitätsmetrik beschreibt die Stabilität der Tests zu einem bestimmten Zeitpunkt. Sowohl die „Pass Rate" als auch der „CPK" allein sind nicht aussagekräftig genug, um eine Aussage über die aktuelle Qualität zu geben. Notwendig ist die Information über den Spezifikationsabdeckungsgrad der definierten Tests. Da im Fallbeispiel das Lastenheft mit diesen Spezifikationen jedoch nicht in das einheitliche Datenformat überführt wurde, wurde die „Teststabilität" als ein weiterer Indikator in der Qualitätsüberwachung verwendet. Das Hinzunehmen resultiert aus der Hypothese, dass neue bzw. sich ändernde Tests auf eine noch nicht fertige bzw. qualitativ noch nicht vollständige Implementierung hinweisen.

$$1 - \frac{\frac{\#newMeasures}{\#totalMeasures} + \frac{\#modifiedMeasures}{\#totalMeasures}}{2}$$

Die Formel berechnet sowohl den prozentualen Anteil neuer Tests als auch den prozentualen Anteil modifizierter Tests (geänderte Spezifikationsgrenzen) und berechnet daraus den Mittelwert. Die beiden Werte werden gleich gewichtet.

Abbildung 56 stellt die Umsetzung ausgewählter Teile des Metrik Baums „Analog Quality“ dar. Derived Measure Beispiele sind die Implementierungen von „Analog Quality“ und „Pass Rate“. Die Abbildung zeigt deren Eingangsmetriken. Base Measure Beispiele sind „Total Tests“ und „Passed Tests“, die beiden Eingangsmetriken von „Pass Rate“. Beide sind jeweils durch eine OCL Query beschrieben.

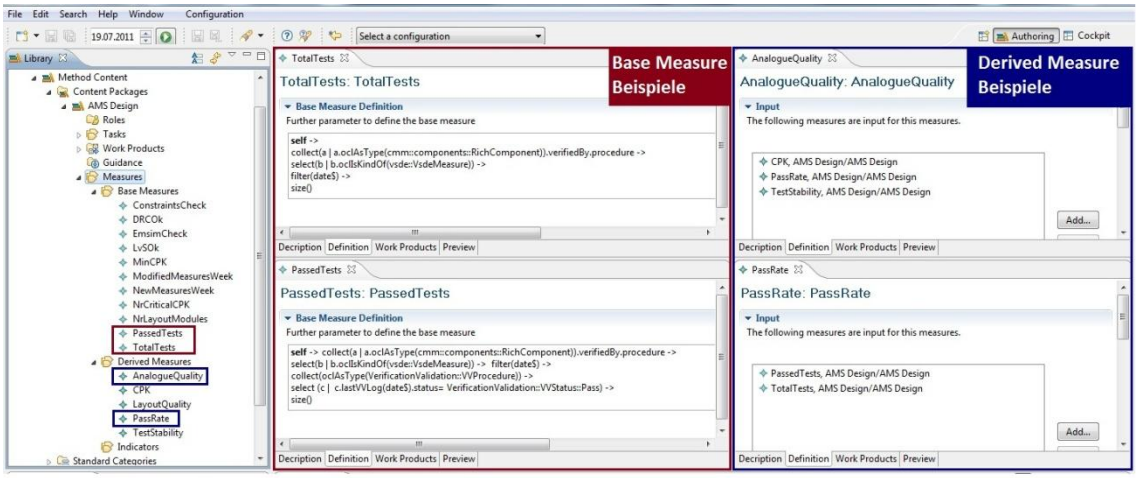

Abbildung 56: Beispiel Metrik Umsetzung

Auf die weitere Darstellung der Metrik Umsetzung wird aus Übersichtsgründen verzichtet. Man sieht jedoch in Abbildung 56 links alle in diesem Abschnitt genannten Qualitätsmetriken.

***Derived Measure:* Layout Qualität**

Work Product: Layout Struktur; ***Task:*** Layout Entwurf & Verifikation

Beschreibung: Analog zum vorherigen Abschnitt beschreibt dieses Kapitel den Aufbau des Qualitätsmodells für den Bereich Analog/Digital Backend. Mögliche Erweiterungen finden sich ebenfalls im abschließenden Kapitel dieses Dokuments. Abbildung 57 stellt den Aufbau der Qualitätsmetrik dar.

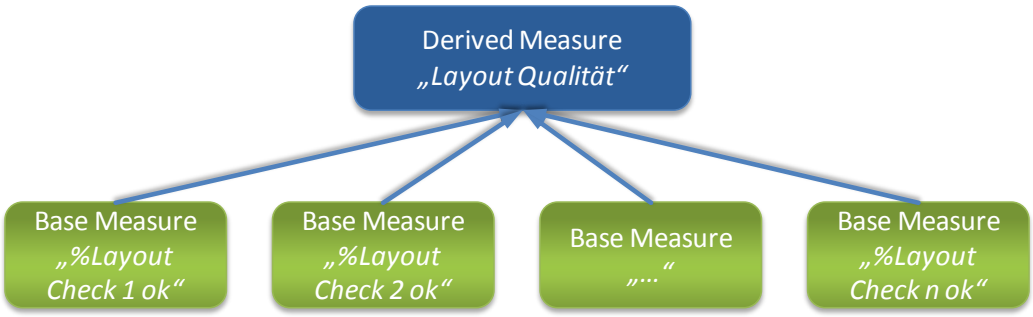

Abbildung 57: Layout Qualitätsmetrik

Die Qualität des Layouts berechnet sich aus dem Mittelwert mehrerer Layout Verifikationschecks, die aus Gründen der Vertraulichkeit an dieser Stelle nicht im Detail beschrieben werden. Jeder Layout Check wird mittels Base Measure gemessen und mit Hilfe eines Derived Measure wie folgt berechnet:

$$\frac{\sum_1^n \%Layout\ Check\ n\ ok}{n}$$

***Derived Measure:* Anforderungserfüllung von Flächenanforderungen**

Work Product: Physikalische Anforderungen; ***Task:*** Layout Entwurf & Verifikation

Beschreibung: Neben den beiden bereits beschriebenen Qualitätsmetriken werden im aktuellen Anwendungsbeispiel formulierte Flächenanforderungen mit Daten aus dem Layout abgeglichen. Die physikalischen Flächenanforderungen sind mit der zugehörigen Komponente verknüpft. Zur Laufzeit werden Zielwerte und Toleranzgrenzen der Flächenanforderungen mit tatsächlichen Werten für diese Komponente aus dem Layout abgeglichen. Denkbar ist hier ebenfalls ein Abgleich mit früheren Flächenabschätzungen, die z.B. schon aus digitalen Synthesetools kommen, welche im Fallbeispiel jedoch nicht betrachtet wurden.

Konfiguration Datenerfassung und -speicherung

Nachdem der zu betrachtende Entwicklungsprozess inkl. Qualitätsmetriken definiert wurde, folgt in diesem Abschnitt die Beschreibung der Abbildung der für die Auswertungen benötigten Entwicklungsdaten auf das Core-Produktmetamodell und der zugehörigen Transformationen zum Laden der Daten aus ihren proprietären Formaten.

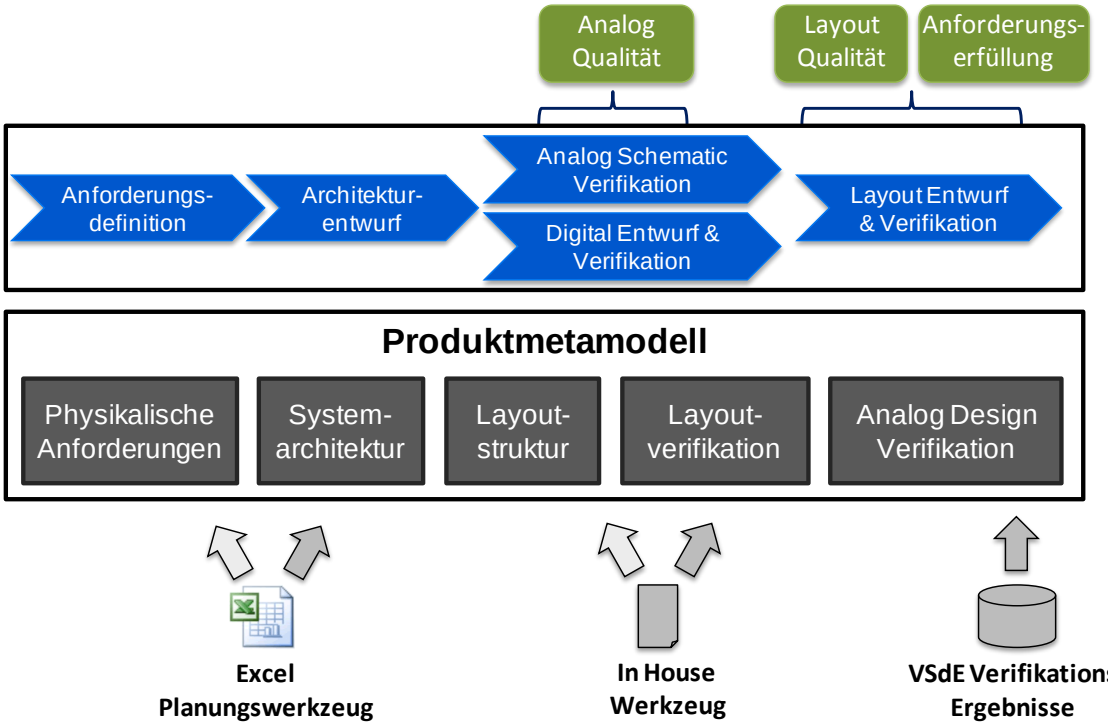

Abbildung 58: Abzubildende Produktdaten und zugehörige Transformationen

Entsprechend definierter In- und Outputs des Entwicklungsprozesses und der definierten Qualitätsmetriken sind im Fallbeispiel fünf Partialmodelle identifiziert worden (siehe Abbildung 58 unten). Entwickelte Transformationen füllen diese Partialmodelle mit Daten:

- Manuelle Transformation von Flächenanforderungen und Systemarchitektur aus einem Excel Sheet in das Metamodell Format.
- Transformation von VSdE (Virtuoso Specification-driven Environment[14] der Firma Cadence[15]) Exporten in das V&V Metamodell.
- Transformation von Daten aus einem Bosch internen In-House Werkzeug auf das Design und V&V Metamodell, da es sowohl die eigentliche Layout Struktur als auch Verifikationsergebnisse enthält.

Der nächste Abschnitt beschreibt die Partialmodelle und deren Abbildung auf die Konzepte des Core-Produktmetamodells. Die Transformationen 2 und 3 sind als Plugins für den entwickelten Prototyp aus Kapitel 4 implementiert.

Physikalische Anforderungen

Für die Beschreibung von physikalischen Anforderungen bzw. der im Fallbeispiel betrachteten Flächenanforderungen ist das Anforderungsmetamodell des Core-Produktmetamodells ausreichend, so dass keine Erweiterungen notwendig sind.

Konzept	Beschreibung	Metamodellkonzept
Flächen-anforderung	Eine Flächenanforderung, die für eine Komponente in der Systemarchitektur die maximale Fläche definiert.	`Constraint` des Anforderungsmetamodells Verweis auf `RequirementKind` „Area“ und einem `Satisfy` Link auf die Komponente.
Zielwert	Die einzuhaltende maximale Fläche.	Ein Attribut am `Constraint`, sowie erwarteter Zielwert einer `VVProcedure`.
Toleranzen	(Optional) Toleranzgrenzen bei der einzuhaltenden Fläche.	Attribute am `Constraint`.

Tabelle 13: Abbildung Physikalische Anforderungen auf Produktmetamodell

[14]Siehe http://www.cadence.com/rl/Resources/datasheets/VirtuosoSpecDriven_ds.pdf

[15]Siehe http://www.cadence.com

System Architektur

Die System Architektur beschreibt eine frühe Dekomposition des Systems in seine einzelnen Komponenten. Die folgende Tabelle beschreibt die einzelnen abzubildenden Konzepte und deren Abbildung auf das Produktmetamodell:

Konzept	Beschreibung	Metamodellkonzept
Component, DigitalComponent, AnalogComponent	Eine Komponente beschreibt einen Teil der Systemarchitektur. Die Systemarchitektur besteht aus Digital- und Analogkomponenten.	`Component`
Hierarchien zwischen Komponenten	Komponenten sind in einer bestimmten Designhierarchie angeordnet, die es abzubilden gilt.	`ComponentProperty`

Tabelle 14: System Architektur und Abbildung auf das Produktmetamodell

Sowohl Flächenanforderungen als auch Systemarchitektur wurden innerhalb eines internen Excel basierten Werkzeugs gepflegt. Auf Grund der geringen Datenmenge, die sich darüber hinaus im betrachteten Zeitraum des Projekts nicht geändert hat, wurden Flächenanforderungen manuell übertragen und per Anforderungseditor direkt im entwickelten Prototyp modelliert. Selbiges gilt für die Systemarchitektur und seiner Komponenten, sowie für die Verknüpfung der Flächenanforderungen an die einzelnen Komponenten.

Analog Schematic Verifikationsergebnisse

Datenquelle der Analog Schematic Verifikationsdaten zur Auswertung der Analog Schematic Qualitätsmetrik ist VSdE von Cadence. Abbildung 59 beschreibt alle innerhalb eines standardisierten VSdE XML Exports enthaltenen Daten:

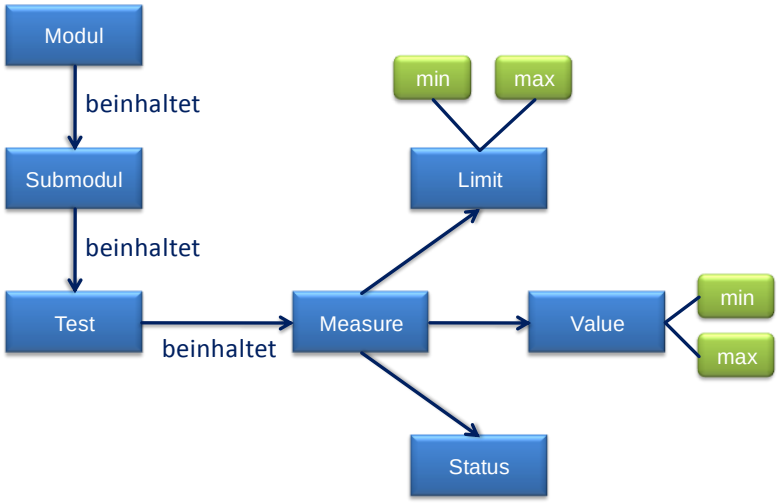

Abbildung 59: Konzepte Analog Frontend Verifikation - VSdE

Die folgende Tabelle beschreibt die einzelnen Konzepte und deren Abbildung auf das V&V Metamodell:

Konzept	Beschreibung	Metamodellkonzept
Modul	Ein Modul besitzt Submodule und wird als Gruppierungselement verwendet.	Keine Abbildung
Submodul	Ein Submodul beinhaltet Tests und wird als Gruppierungselement verwendet.	Keine Abbildung
Test	Ein Test beinhaltet mehrere Measures.	`VerificationCase`
Measure	Ein Measure beschreibt einen konkreten Test, inklusive Zielwerte und Ergebnisse. Auf Basis dieser Werte wird der CPK Wert berechnet. Anmerkung: Nicht zu verwechseln mit dem Konzept `MeasureDefinition`!	`Verification-Procedure`
Limit	Spezifikation der Zielwerte des Tests bestehend aus einem minimalen und einem maximalen Zielwert.	`VVIntendedOutcome` der `VerificationProcedure` repräsentiert durch eine Subklasse von `VVOutcome`
Value	Ergebnis des Tests bestehend aus einen maximalen und einem minimalen Wert.	`VVLog` mit einem `VVActualOutcome` zur Beschreibung des Testergebnisses
Status	Eine Testbench testet entweder das Schematic oder das Layout. Dies wird durch die Zuordnung einer Cell Instanz zu einer View beschrieben.	`VVStatus`

Tabelle 15: Abbildung Analog Frontend Konzepte auf V&V Metamodell

Das Analog/Digital Layout beschreibt die Hierarchie des Layouts. Darüber hinaus sind für jede Zelle Metadaten über Layout spezifische Verifikationsschritte annotiert. Abbildung 60 visualisiert die Daten, die in einem XML Export eines bei Bosch intern entwickelten In-House Werkzeugs vorliegen:

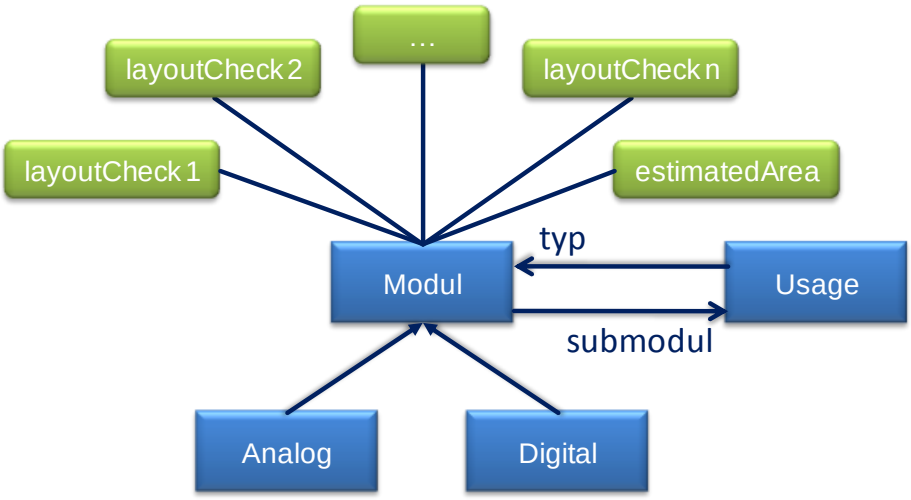

Abbildung 60: Konzepte Layout Struktur und Verifikation

Die folgende Tabelle beschreibt die einzelnen Konzepte der Layout Struktur und dessen Verifikation, sowie deren Abbildung auf das Produktmetamodell:

Konzept	Beschreibung	Produktmetamodell
Modul, Analog, Digital	Ein Modul steht für eine Zelle im Layout. Diese ist entweder Analog oder Digital.	`Component`
Usage	Verknüpft zwei Zellen miteinander. Beispiel: Modul1 → hasSub → Usage → isSub → Modul2. Dies definiert, dass Modul2 eine Unterkomponente von Modul1 ist.	`ComponentProperty`
layoutCheck 1, layoutCheck 2, layoutCheck n	Jede Zelle besitzt Informationen, ob bestimmte Verifikationschecks erfolgreich waren. Jeder dieser Checks ist abzubilden.	Jede Zelle ist mit einem `VerificationCase` verknüpft (`VVKind` „Layout Verification“), welcher je einer `VerificationProcedure` pro Check besitzt. Ergebnisse der Checks werden mit `VVLog` abgespeichert.
areaEstimation	Beschreibt das Ergebnis einer Flächenabschätzung für jede Zelle.	Jede Zelle besitzt einen `VerificationCase` mit einer `VerificationProcedure`. `VVLog` spei-

		chert für diese `VerificationProcedure` den konkreten Flächenwert für die Zelle unter `VVActualOutcome`.

Tabelle 16: Betrachtete Konzepte des Analog/Digital Layout

Durch die Abbildung auf das Core-Produktmetamodell stehen darüber hinaus alle dort definierten Konzepte zur Verfügung, um Flächenanforderungen mit der Systemarchitektur zu verbinden und die Analogtests an Zellen aus dem Schematic zu verknüpfen. Um Komponenten der Systemarchitektur mit den Zellen aus dem Layout zu verknüpfen, wurde eine weitere Verknüpfung mittels einer Metamodell Erweiterung eingefügt (`Realize` Relation). So erzeugt das Produktmetamodell eine integrierte Sicht auf ehemals verteilte Produktdaten.

Prozessinstanziierung

Nachdem das Qualitätsmodellierungsframework entsprechend dem Entwicklungsprozess und der Qualitätsdefinition kalibriert ist, wurden die definierten Prozessbausteine mit Hilfe des Wizards aus Abschnitt 4.2.3 für ein ausgewähltes AMS Projekt instanziiert. Ausgangspunkt war dabei eine bereits vollständig entwickelte Systemarchitektur. Aus Gründen der Vertraulichkeit kann an dieser Stelle die konkrete Systemarchitektur nicht genannt werden. Für jede Analog und Mixed Signal Komponente wurden, wie in Abbildung 61 dargestellt, die Tasks „Layout Entwurf & Verifikation“ und „Analog Schematic Verifikation“ instanziiert. Jede definierte Qualitätsmetrik erhält somit ihren Kontext, gegen den sie ausgewertet wird. Dies erlaubt eine detaillierte Qualitätsauswertung auf Komponentenebene.

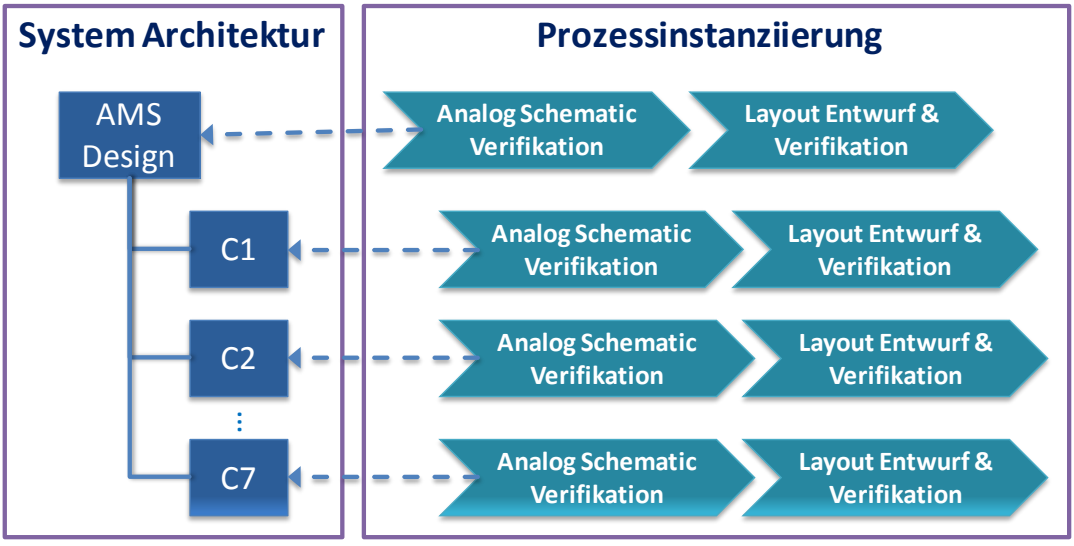

Abbildung 61: Beispielhafte Prozessinstanziierung für die AMS Systemarchitektur

Die Prozessinstanziierung diente innerhalb des Anwendungsbeispiels als Vorlage zur Erzeugung eines Projektplans. Dieser ließ sich mit MS Project importieren und mit Start- und Endterminen, Ressourcen und einer konkreten Ablaufplanung verfeinern. Die Planung wurde laufend mit der internen Repräsentation des Prototyps synchronisiert, um den Projektkontext bei der Qualitätsbewertung zu berücksichtigen: Wann soll welche Task für welche Komponente fertiggestellt sein? Existieren Zwischenmeilensteine, an denen ein bestimmter Wert für die Metriken erreicht sein muss?

Qualitätsbewertung

Der Prototyp annotierte zur Laufzeit regelmäßig aktuelle Qualitätsstände an die definierten Tasks des Projektplans und verglich diesen Ist-Stand mit Soll-Werten aus der Projektplanung. Eine von Bosch im Projekt PRODUKTIV+ entwickelte Projektplanoptimierungskomponente (Blaschke, 2009a), (Blaschke, 2009b), (Blaschke, 2010) nahm bei Planabweichungen Anpassungen vor. Das Zusammenspiel zwischen Qualitätsbewertung und Projektplanoptimierung ist im Detail in (Häusler, 2009) und (Häusler, 2010) beschrieben. Die folgenden Abschnitte stellen die durchgeführten Auswertungen exemplarisch dar.

Die *Design Hierarchy View* in Abbildung 62 zeigt die Ergebnisse der Qualitätsmetriken „Layout Qualität“ und „Analog Qualität“ für eine Komponente. Jede Metrik lässt sich bis zu ihren messbaren Bestandteilen verfolgen. So sind der Komponente z.B. vier Layout Zellen zugeordnet (siehe „Number of realized Layout Modules“). Bis auf den ersten Check waren alle notwendigen Analysen erfolgreich (siehe 0.0 Wert in Abbildung 62 für den ersten Layout Check). Auf dieselbe Weise lassen sich für alle anderen Komponenten die Ergebnisse detailliert betrachten.

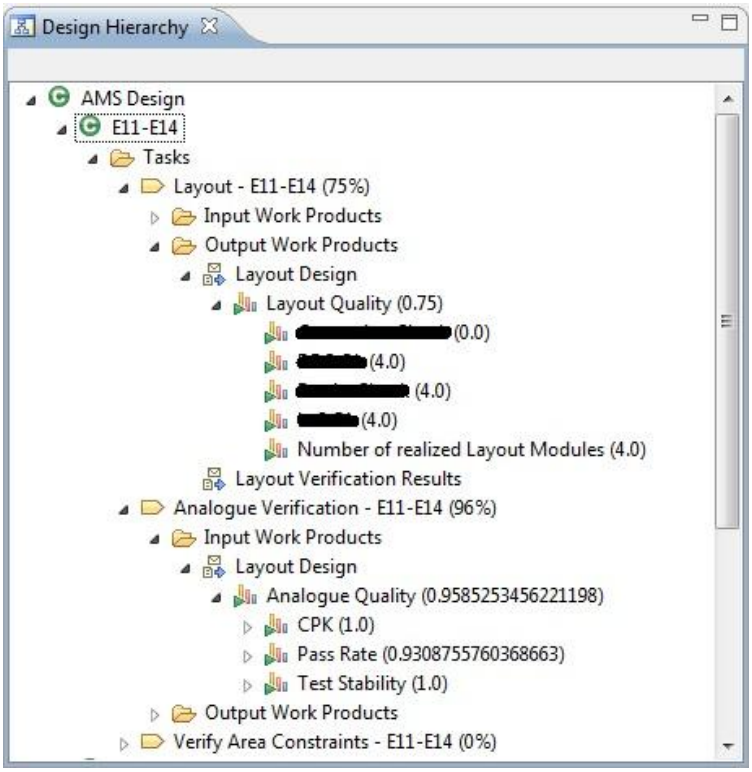

Abbildung 62: Übersicht Qualitätsstand für eine Komponente

Alle Auswertungen lassen sich entwicklungsbegleitend in regelmäßigen selbst gewählten Abständen wiederholen, so dass der Wertverlauf über die Zeit verfolgbar ist. Exemplarisch ist dies für den Gesamtverlauf der „Layout Qualität“ in Abbildung 63 dargestellt.

Abbildung 63: Verlauf Layout Qualität (exemplarisch)

Diese beiden exemplarischen Einblicke in die Qualitätsbewertung demonstrieren die automatischen und entwicklungsbegleitenden Auswertungen definierter Qualitätsmetriken.

5.1.4 Ergebnisse

Die entwickelte Methodik und das zugehörige Qualitätsmodellierungsframework adressieren erfolgreich die eingangs dieses Anwendungsbeispiels formulierten Ziele. Das Produktmetamodell bildet alle gewünschten Produktstrukturen und Verifikationsergebnisse ab, um die durch Entwickler ausgewählten Qualitätsmetriken auszuwerten. Hinzugefügt wurde lediglich eine spezielle Verknüpfung zwischen Konzepten der System Architektur und Zellen des Layout Modells. Die prozessorientierte Definition von Qualitätsmetriken erlaubte darüber hinaus, eine entwicklungsbegleitende Verfolgung der Produktqualität auf Komponentenebene und eine Operationalisierung der Qualitätsbewertungen für die Projektsteuerung. Die gesamte Anwendung beschreiben die Veröffentlichungen (Häusler, 2009) und (Häusler, 2010). Die Zielerfüllung lässt sich wie folgt zusammenfassen

- **Ziel 1: Einheitliche Qualitätsdefinition:** Alle mit den Entwicklern ausgewählten Qualitätsmetriken wurden mit Hilfe der Konzepte im Prozessmetamodell abgebildet. Das Ziel wurde **erfüllt**.

- **Ziel 3: Integration V&V Ergebnisse:** Im vorherigen Kapitel wurde demonstriert, wie alle ursprünglich verteilt vorliegenden V&V Ergebnisse auf das Produktmetamodell abgebildet wurden. Das Ziel wurde **erfüllt**.
- **Ziel 5: Entwicklungsbegleitende Auswertung:** Wie an einem Beispiel gezeigt, ist das Qualitätsmodellierungsframework prinzipiell in der Lage, eine entwicklungsbegleitende Auswertung durchzuführen. Datenerfassung und –analyse ist an beliebigen Zeitpunkten der Entwicklung initialisierbar. Das Ziel wurde **erfüllt**.
- **Ziel 6: Berücksichtigung von Prozess- und Projektkontext:** Die regelmäßige Annotation von Qualitätsbewertungen an die verschiedenen Tasks ermöglicht deren Fortschrittsbewertung. Durch den Metamodell basierten Ansatz ist eine leichte Anbindung beliebiger Projektmanagement- und Planungsoptimierungswerkzeuge gegeben, um den Projektkontext, d.h. Plandaten zu berücksichtigen. Das Ziel wurde **erfüllt**.
- **Ziel 7: Automatische Datenerfassung und –integration:** Sowohl Datenerfassung mittels Modelltransformationen als auch die Datenanalyse durch die formalisierten Metriken ließen sich vollständig automatisieren. Das Ziel wurde **erfüllt**.
- **Ziel 8: Anpassbarkeit an bestehende Entwicklungsprozesse:** Es wurden alle für die Bewertung notwendigen Datenquellen angebunden. Eine Qualitätsbewertung erfolgt prinzipiell ohne Entwickler bei ihrer täglichen Arbeit zu beeinflussen. Prinzipiell, da das betrachtete Projekt bereits abgeschlossen war. Letztendlich ist dieses Ziel nur in einem tatsächlich laufenden Projekt mit anschließender Entwicklerbefragung bewertbar. Das Ziel wurde **teilweise erfüllt**.

Obwohl sich festhalten lässt, dass die Mehrzahl der zu Beginn des Anwendungsbeispiels formulierten Ziele im Rahmen einer komplexen industriellen Entwicklungsumgebung erfüllt wurde, besteht auch hier Verbesserungspotential. Im Anwendungsbeispiel war eine Überführung von Spezifikationen für die Analogkomponenten in das einheitliche Datenformat nicht möglich, was sich gerade bei der Bewertung des Verifikationsgrads eben dieser negativ auswirkte. Hohe Werte für „Pass Rate“ und „CPK“ bedeuten nicht automatisch eine hohe Qualität. Zu Beginn des Projekts fehlt z.B. noch eine Vielzahl an Tests, eine hohe „Pass Rate“ hat demnach nur eine eingeschränkte Aussagekraft. Der Stabilitätsindikator schafft zwar zum Teil Abhilfe, ist jedoch fehlerbehaftet, wenn sich über einen längeren Zeitraum wenig bis gar nichts ändert. Zur Verbesserung der Produktqualitätsbewertung ist eine Aussage über die Erfüllung der Spezifikation zu treffen. Für diese Aussage müssen die einzelnen Spezifikationspunkte maschinenlesbar zur Verfügung gestellt und mit zugehörigen Tests verknüpft werden. Dieser Integrationsschritt ermöglicht die Überprüfung, ob ein oder mehrere zugeordnete Tests bereits die geforderten Spezifikationsgrenzen umsetzen bzw. zu wie viel Prozent ein Spezifikationspunkt erfüllt ist. Eine Erkenntnis, welche die Argumentation dieser Arbeit untermauert.

5.2 Anwendungsbeispiel sicherheitskritische eingebettete Systeme

Das zweite Anwendungsbeispiel evaluiert die Methodik im Kontext der ISO26262, dem Sicherheitsstandard für die Entwicklung von Automobilelektronik. Die Evaluation prüft die Anwendbarkeit der Methodik für das Management der funktionalen Sicherheit in der System Design Phase (ISO26262 - Kapitel 4). Das im BMBF geförderten europäischen Verbundprojekt CESAR entstandene Anwendungsbeispiel wurde in Teilen in (Jost, 2010) veröffentlicht.

5.2.1 Anwendungskontext

Elektronik bzw. eingebettete Systeme sind elementarer Bestandteil heutiger Automobile und prägen auch zukünftig die Entwicklung neuer Funktionalitäten und Innovationen. Bereits 2005 waren laut einer Studie (Fast, 2005) 90% aller Innovationen innerhalb des Automobils IKT getrieben. Beispiel hierfür sind Sicherheitssysteme wie der Airbag, die Antischlupfregelung oder die Elektronische Stabilitätskontrolle. Aber auch Fahrassistenzsysteme die zur Minimierung von Unfallrisiken noch aktiver in das Fahrverhalten eingreifen, finden immer mehr ihren Weg in das Automobil. Dieser Trend hat sowohl Vor- als auch Nachteile. Auf der einen Seite erhöht dies die Sicherheit aktueller Fahrzeuge. Auf der anderen Seite trägt es aber auch zur Komplexitätssteigerung bei, die vor allem die Entwicklung handhaben muss. Bereits heute enthalten Fahrzeuge 80 und mehr ECU's (Electronic Control Unit) und eine noch größere Anzahl an Sensoren und Aktuatoren (Fraunhofer, 2011). Funktionen sind immer stärker integriert, denn erst durch die Kombination an Sensoren, Aktuatoren und Software entstehen neue Funktionen und Innovationen (ForTISS, 2011). Dieser Trend ist dabei noch keinesfalls am Ende. Themen wie die Elektromobilität und neue dafür notwendige IKT Architekturen sind nur ein Beispiel (ForTISS, 2011).

Auf Grund ihrer Rolle im Automobil und den in sie gestellten Qualitätsanforderungen hinsichtlich Ausfallsicherheit, sind in den letzten Jahren die normativen Anforderungen an die Entwicklung sicherheitskritischer Automobilelektronik gestiegen. Diese Anforderungen manifestieren sich in der ISO26262, dem Standard für die Entwicklung sicherheitskritischer Elektronik im Automobil. Verabschiedet wird dieser voraussichtlich 2012. Je nach Kritikalität des Systems stellt dieser bestimmte Anforderungen an Entwurf- und Verifikationsprozess. Entwicklungsabteilungen müssen sich zukünftig nach diesem Standard richten und nachweisen, dass ihre Systeme ausreichend ausfallsicher sind. Eine Methodik, welche bereits während der Entwicklung prüft, ob vorgeschriebene Schritte des Standards durchgeführt wurden, stellt einen Beitrag für diese Herausforderung dar.

5.2.2 Ziele

Analog zum Vorgehen im ersten Anwendungsbeispiel untersuchte die Evaluation die Erfüllung von Zielen dieser Arbeit am Gegenstand eines funktionalen Sicherheitsmanagements der ISO26262 System Design Phase. Der folgende Abschnitt nennt die unter-

suchten Fragestellungen, die jeweils mit einem Ziel der Gesamtzielstellung (siehe Abschnitt 1.4) verknüpft sind:

1) Inwieweit eignet sich das Qualitätsmodellierungsframework zur Umsetzung einer einheitlichen ISO26262 konformen Qualitätsdefinition, um die Umsetzung von Tasks der ISO26262 System Design Phase zu überwachen? → **Ziel 1: Einheitliche Qualitätsdefinition.**

2) Der ISO26262 Referenzprozess enthält abhängig von der Risikoklasse des Systems unterschiedliche Ausprägungen (z.B. bezüglich anzuwendender Verifikationsmethoden). Die Risikoklasse eines Systems und seiner Teilkomponenten wirkt sich auf Entwicklungsprozess und anzuwendende Qualitätsdefinition innerhalb eines Projekts aus. Eignet sich das Modellierungssframework für die Abbildung einer projektübergreifenden Qualitätsdefinition und deren effizienten Kalibrierung auf ein Projekt? → **Ziel 2: Kalibrierung auf einzelnes Projekt.**

3) Eignet sich das Qualitätsmodellierungsframework zur Abbildung aller anfallenden Verifikationsergebnisse des System Designs, insbesondere zur Abbildung aller Ergebnisse der Anforderungsvalidierung, die gerade in der System Design Phase eine zentrale Rolle spielt? → **Ziel 3: Integration V&V Ergebnisse.**

4) Der ISO26262 Referenzprozess fordert je nach ASIL Klassifikation verschiedene Entwicklungsmethoden. Nur wenn die Entwicklung einem empfohlenen Prozess folgt, also eine gewisse Prozessqualität einhält, ist nach ISO26262 ein bestimmter Qualitätsstandard eingehalten. Ist es mit dem Qualitätsmodellierungsframework möglich, auch solche prozessbezogenen Qualitätsbewertungen umzusetzen, wie es für das Management funktionaler Sicherheit nach ISO26262 notwendig ist? → **Ziel 6: Berücksichtigung von Prozess- und Projektkontext.**

Der folgende Abschnitt beschreibt die Umsetzung des Anwendungsbeispiels im Detail.

5.2.3 Durchführung

Analog zum ersten Beispiel folgt die Durchführung dem in Abschnitt 3.3 beschriebenen Vorgehensmodell und wird entlang den dort definierten Schritten beschrieben.

Prozessdefinition

Grundlage für ein prozessorientiertes Qualitätsmonitoring gemäß ISO26262 ist der durch den Standard vorgeschlagene Referenzprozess. Die ISO26262 enthält Anforderungen und Empfehlungen für die Produktentwicklung. Dies sind sowohl Anforderungen an das Produkt (z.B. Anforderungen an die Ausfallwahrscheinlichkeit von Hardware Komponenten), aber auch an den Entwicklungsprozess selbst. Um die Erfüllung dieser Prozessanforderungen und damit eine bestimmte Prozessqualität während der Erfüllung mit Hilfe des entwickelten Frameworks nachzuweisen, ist ein Prozessmodell zu erstellen. Dieser Abschnitt beschreibt dieses Prozessmodell für drei Kapitel des vierten Teils des Standards (Product Development: System Level):

- Kapitel 5: „Initiation of product development at the system level"

- Kapitel 6: „Specification of technical safety requirements"
- Kapitel 7: „System design"

Grundlage der Prozessdefinition ist ein bereits im CESAR Projekt entwickeltes auf der ISO26262 basierendes Framework für die funktionale Sicherheit in der Automobilelektronik. An der Definition waren Partner wie Fiat, Volvo, Infineon, AVL und Delphi beteiligt, so dass die Grundlage hinreichend durch industrielle Akzeptanz abgesichert ist. Eine erste Version dieses Frameworks beschreibt Krammer (Krammer, 2011). Folgende Tasks wurden aus der ISO26262 extrahiert und modelliert:

Initiation of product development at the system level	**Specification of technical safety requirements**	**System Design**
Planning of functional safety assessment	Specify technical safety requirements (informal)	Allocate requirements to hardware and software
Planning of item integration and testing	Specify technical safety requirements (semi-formal)	Analyse causes and effects of systematic failures
Planning of product development at system level	Specify technical safety requirements (formal)	Define measures for control of random hardware failures
Planning of validation of activities	Identify multiple point faults	Perform deductive safety analysis
	Specify safety mechanisms to detect multiple point faults	Perform inductive safety analysis
	Validate informal technical safety requirements	Specify hardware software interfaces
	Validate semi-formal technical safety requirements	Specify integration tests
	Validate formal technical safety requirements	Specify system design and technical safety concept
	Specify validation criteria	System design inspection
	Specify safety related requirements concerning production, maintenance, repair and decomissioning	System design simulation

		System design walkthrough
		System prototyping and vehicle tests

Tabelle 17: Modellierte Tasks der ISO26262

Einige Tasks sind im Vergleich zum CESAR Framework für funktionale Sicherheit detaillierter modelliert. Dabei wurde Wert darauf gelegt, auch alle variablen Prozessschritte und Methoden explizit als Tasks zu modellieren. Jede Methode aus Tabelle 18 wird zu einem entsprechenden Task im Prozessmodell.

Methods	**ASIL**			
	A	**B**	**C**	**D**
System design inspection	+	++	++	++
System design walkthrough	++	+	o	o
System design simulation	+	+	++	++
System design prototyping and vehicle tests	+	+	++	++
Deductive analysis	o	o	+	++
Inductive analysis	++	++	++	++

Tabelle 18: System Design Verification aus ISO26262

Der Standard gibt für jede Methode entsprechend der ASIL Klassifikation des betrachteten Systems eine unterschiedlich starke Empfehlung für ihre Verwendung, was sich neben der System Design Verifikation auch an anderen Stellen im Standard wiederfindet. Die Arbeit verwendet zur Prozessdefinition dabei folgende Interpretation der Empfehlungen:

- *(++) – Die Methode wird für den ASIL stark empfohlen* → Verwendung der Methode in der Prozessdefinition
- *(+) – Die Methode wird für den ASIL empfohlen* → Optionale Verwendung der Methode in der Prozessdefinition. Auswahl während Prozessinstanziierung
- *(o) – Die Methode hat keine Empfehlung für oder gegen ihre Anwendung für den ASIL* → Keine Verwendung der Methode in der Prozessdefinition.

Diese Interpretation ist dabei lediglich in dieser Arbeit maßgebend. Grundsätzlich ist es jedem Unternehmen selbst überlassen, die Empfehlungen auf andere Weise zu interpretieren. Eine Anpassung der Prozessdefinitionen erlaubt dies ohne größeren Aufwand.

Basierend auf der Interpretation wurden Prozesse für die ASIL Klassen A, B und C erstellt. Abbildung 64 zeigt, wie mit steigender Kritikalität a) die Anforderungen stärker formalisiert werden, um Qualitätseigenschaften von Anforderungen auch automatisch abzusichern und b) wie sich die Anzahl geforderter Verifikationsmethoden ändert. Es ist anzumerken, dass die ISO26262 die Beschreibung formaler Anforderungen bestenfalls mit einem (+) versieht. Trotzdem ist dies im ASIL C Prozess als verpflichtend modelliert, um die Variabilität der Beispielprozesse zu erhöhen.

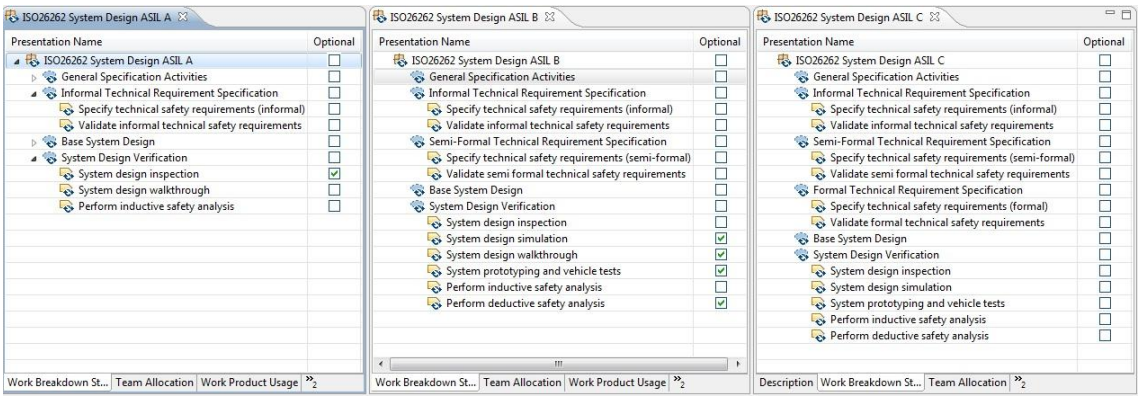

Abbildung 64: Definierte ISO26262 Prozesse

Basierend auf den definierten Methoden und Prozessen folgt die Beschreibung der Qualitätsdefinition.

Prozessorientierte Qualitätsdefinition

Anforderungen sind einer der zentralen Aspekte innerhalb der ISO26262, speziell in der Phase des System Designs, so dass sich für das Qualitätsmonitoring eines ISO26262 basierten Entwicklungsprozesses eine anforderungsorientierte Bewertung eignet. Der folgende Absatz beschreibt die Messung von Anforderungsqualität und –erfüllung als zentrale Qualitätsindikatoren innerhalb des Anwendungsbeispiels:

Anforderungsqualität

Eines der primären Ziele der System Design Phase ist die Entwicklung möglichst vollständiger und konsistenter technischer Sicherheitsanforderungen. Die Evaluation verwendet diverse Kriterien zur Messung von Anforderungsqualität aus Abschnitt 2.3.1, verfeinert diese und annotiert sie an Tasks des Prozessmodells.

Wie in Abbildung 65 dargestellt, detailliert das Modell vier Qualitätscharakteristika. Jedes dieser Qualitätscharakteristika ist wiederum in einzelne Qualitätsattribute unterteilt, von denen blau dargestellte Attribute verpflichtend und grüne optional sind. Jedes dieser Qualitätscharakteristika wird in Form von Qualitätsmetriken beschrieben und einzelnen Tasks zugeordnet. Die Zahlen an den Attributen visualisieren diese Zuordnung (siehe auch Abbildung 66). Anmerkung: Die Arbeit liefert keine vollständige Definition von Anforderungsqualität, sondern demonstriert vielmehr, wie eine prozessorientierte Qualitätsdefinition für die Kalibrierung eines Qualitätsmodell mit optionalen Elementen auf einen konkreten Projektkontext nutzbar ist.

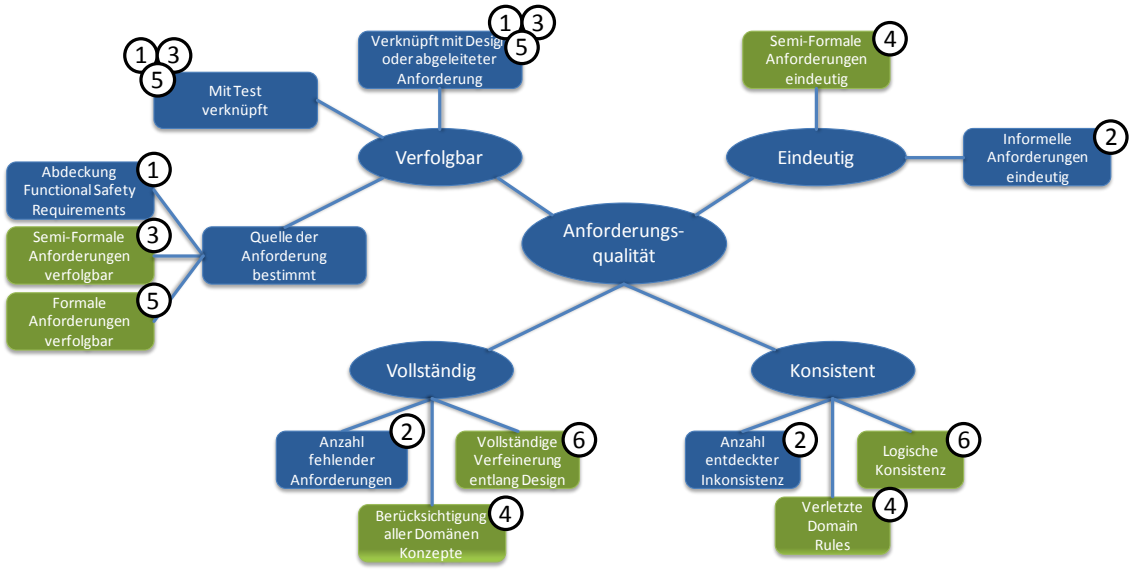

Abbildung 65: Beispielhaftes Modell für Anforderungsqualität

Die Bewertung berücksichtigt die optionalen Teile lediglich, wenn Anforderungsformalisierungsschritte durchgeführt werden. Durch die Zuordnung an einzelne Prozessschritte führt die Prozesskalibrierung für eine Komponente gleichzeitig zur Kalibrierung des Qualitätsmodells. Beispiel: Eine ASIL A Komponente verlangt keine Anforderungsformalisierung in einer semi-formalen Notation, weshalb die Bewertung die den Schritten drei und vier zugeordneten Qualitätsmetriken nicht berücksichtigt.

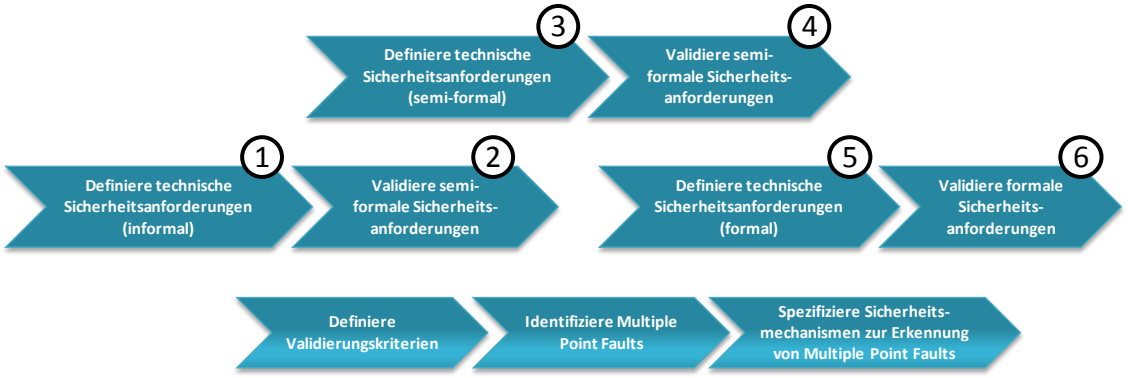

Abbildung 66: Zuordnung der Qualitätsmetriken an Prozessschritte

Anmerkung: Den drei unten dargestellten Tasks wurden keine Qualitätsmetriken zugeordnet. „Specify validation criteria“ deutet lediglich an, dass parallel zur Anforderungsdefinition die Definition von Integrationstests gestartet werden sollen. Hierfür gibt es in der ISO26262 einen eigenen Abschnitt, der im Anwendungsbeispiel nicht betrachtet wurde. Die anderen beiden Schritte umfassen die Durchführung von Fault Tree Analysen zur Identifikation von Multiple Point Faults, was im Anwendungsbeispiel ebenfalls nicht berücksichtigt wurde.

Anforderungserfüllung

Die zentrale Qualitätsmetrik für die Tasks der System Verifikation aus Tabelle 17 ist die Anforderungserfüllung:

- System design inspection
- System design walkthrough
- System design simulation
- System design prototyping and vehicle tests

Für alle Tasks berechnet die Metrik das Verhältnis zwischen der Gesamtanzahl an Sicherheitsanforderungen und der Anzahl erfüllter Sicherheitsanforderungen (siehe Abbildung 67). Der Anforderungserfüllungsgrad berechnet sich gemäß der Definition aus Abschnitt 3.2.1 basierend auf Ergebnissen aller mit der Anforderung verknüpften Tests. Die Implementierung dieser Metrik für die verschiedenen Tasks unterscheidet sich nur in genau einem Filter Parameter, dem Verifikationstyp, dargestellt durch <XX> in Abbildung 67.

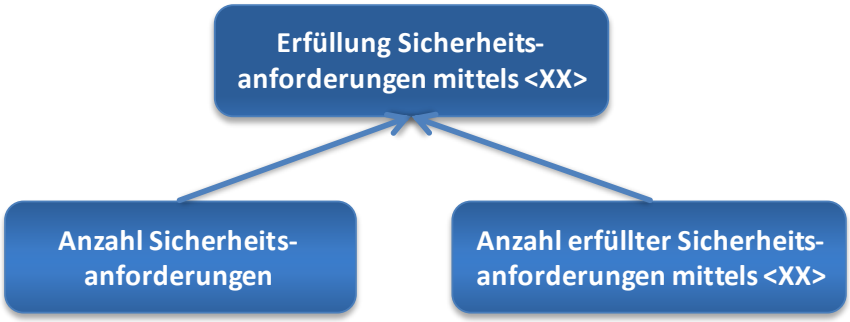

Abbildung 67: Anforderungserfüllungsmetrik

Für die System Design Simulation Task berücksichtigt die Anforderungserfüllungsmetrik z.B. nur die Ergebnisse eines eingesetzten Simulationswerkzeugs. Wenn auch eine Verifikation durch Prototyping gefordert ist, gilt eine Anforderung trotzdem unter diesem Gesichtspunkt als nicht erfüllt. Dies ist ein einfaches aber effektives Beispiel, wie die Methodik letztendlich auch Prozessanforderungen bei der Auswertung berücksichtigt. Eine Anforderung gilt nur als erfüllt, wenn auch alle geforderten Verifikationsmethoden angewandt wurden.

Nicht für alle Tasks wurden Metriken umgesetzt. Für den Task „Allocate requirements to hardware and software“ wäre dies eine einfache Überprüfung, ob alle Anforderungen mittels `Satisfy` einer Hardware oder Software Komponente zugeordnet sind. Die noch übrigen Tasks behandeln im Wesentlichen die Durchführung von Sicherheitsanalysen und die Umsetzung von Mechanismen zur Minimierung nicht akzeptabler Risiken potentieller Fehler, was aber in diesem Abschnitt wie schon bei der Anforderungsqualität nicht betrachtet wurde.

Kalibrierung Datenerfassung und -speicherung

Grundlage des Anwendungsbeispiels ist die in CESAR entwickelte Interoperabilitätslösung. Von verschiedenen Projektpartnern wurden Adapter entwickelt, um Daten aus proprietären Datenformaten zu laden und in das CESAR Metamodell (CMM) zu überführen, welches dabei vollständig kompatibel zu dem in dieser Arbeit entwickelten Core-Produktmetamodell ist. Datenaustausch und –speicherung erfolgt mittels des ModelBus Repositories. Vergleicht man diese Lösung mit dem in Kapitel 4 beschriebenen Prototyp, so übernimmt die in CESAR entwickelte Infrastruktur die Rolle der Erfassung von Entwicklungsdaten. Die restlichen Komponenten des Prototyps wurden an die CESAR Infrastruktur angebunden.

Zur Durchführung des skizzierten Anforderungsformalisierungsprozesses standen folgende Werkzeuge zur Verfügung, die ihre Ergebnisse alle im CMM Format im ModelBus Repository ablegen:

- IBM Rational DOORS zur Definition informeller Anforderungen
- DOTD, ein Forschungsprototyp von Infineon zur Beschreibung semi-formaler Anforderungen in Form von Boilerplates (Farfeleder, 2011)
- Pattern Editor zur Beschreibung formaler Anforderungen (CESAR, 2010d)

Das Produktmetamodell beschreibt Anforderungen und zugehörige Spezifikationselemente (Text, Boilerplate, Automat). Boilerplates sind mit informellen Anforderungen und Patterns mit den jeweiligen Boilerplates mittels `Refine` Relation verknüpft, um die Quelle aller formalen Anforderungen eindeutig zu identifizieren.

Um nun diesen Anforderungsformalisierungsprozess zu überwachen und die definierten Qualitätsmetriken auszuwerten, werden die Analyseergebnisse der Anforderungsvalidierung erfasst und im Produktmetamodell Format abgelegt. Dieser Abschnitt beschreibt dies entlang der vier aufgestellten Qualitätscharakteristika der Anforderungsqualität in Abbildung 65.

Eindeutig

Eine Eindeutigkeitsanalyse wird sowohl anhand der informellen Anforderungen (z.B. in einem manuellen Review) als auch anhand der Boilerplates durchgeführt. Das DODT Werkzeug untersucht den Text auf unerlaubte Schlüsselwörter und prüft, ob verwendete Begriffe zweideutig sind, d.h. Synonyme mit unterschiedlichen Interpretationen existieren. Jeder Anforderung ist eine getypte `ValidationProcedure` (siehe Abbildung 68) zugeordnet, welche mittels `VVLog` das Ergebnis der Analyse speichert.

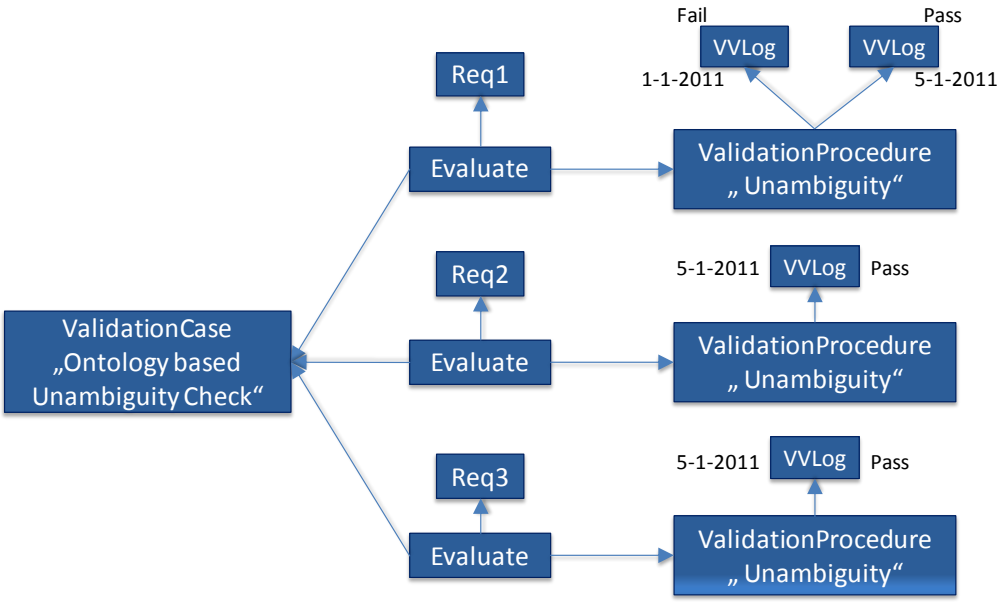

Abbildung 68: Repräsentation der Ergebnisse von Eindeutigkeitsanalysen

Eine Eindeutigkeitsanalyse der informellen Anforderungen mittels Review wird analog abgespeichert und lediglich anders getypt.

Konsistent

Die Konsistenzanalyse wird für alle Anforderungsformalismen durchgeführt. Lediglich die Methode unterscheidet sich. Ob nun manuelles Review, Ontologie basierte Analyse oder formale Analyse, abgespeichert wird die Analyse mittels einem `ValidationCase` und einer `ValidationProcedure`, die mit allen Anforderungen des jeweiligen Formalismus verknüpft ist. Das Ergebnis der Analyse (`RequirementOutcome`) markiert alle Anforderungen, die zueinander inkonsistent sind. `RequirementOutcome` ist eine Spezialisierung des Konzepts `VVActualOutcome` (siehe 3.2.1)

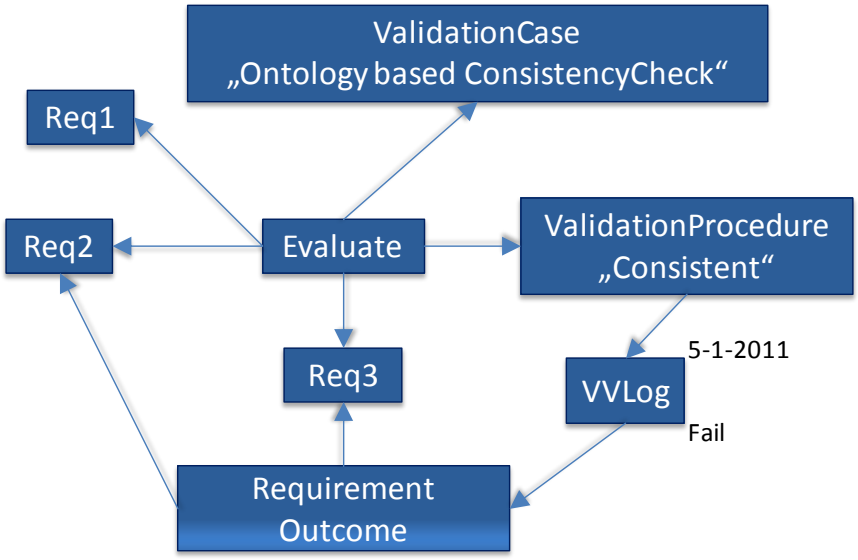

Abbildung 69: Repräsentation einer Konsistenzanalyse

Auch hier gilt: Alle anderen Konsistenzanalysen werden auf dieselbe Weise abgespeichert, nur anders getypt.

Vollständig

Eine Vollständigkeitsanalyse ist Bestandteil der Validierungsmaßnahmen jedes Anforderungsformalismus. Vollständigkeit besitzt diverse Definitionen, die weder im Rahmen von CESAR, noch in diesem Anwendungsbeispiel komplett betrachtet werden. Für jeden Anforderungsformalismus sind beispielhafte Vollständigkeitsanalysen erfasst.

Die Analysen prüfen textuelle Anforderungen und Boilerplates auf Abdeckung aller notwendigen Domänenkonzepte. Ein manuelles Review der informellen Anforderungen fußt auf Expertenwissen. Der Reviewer untersucht die definierten Anforderungen auf fehlende Aspekte. Boilerplates führen diese Analyse mit Hilfe von Ontologien durch, welche entsprechendes Wissen modellieren. Die Pattern basierte Vollständigkeitsanalyse prüft schließlich, ob die Verfeinerung formaler Anforderungen entlang der System Architektur vollständig ist. Eine `ValidationProcedure` pro Anforderung beschreibt das Ergebnis der Vollständigkeitsanalyse (siehe Abbildung 70). Eine fehlgeschlagene Analyse deutet auf fehlende Aspekte bei der Anforderung bzw. ihrer Verfeinerung hin.

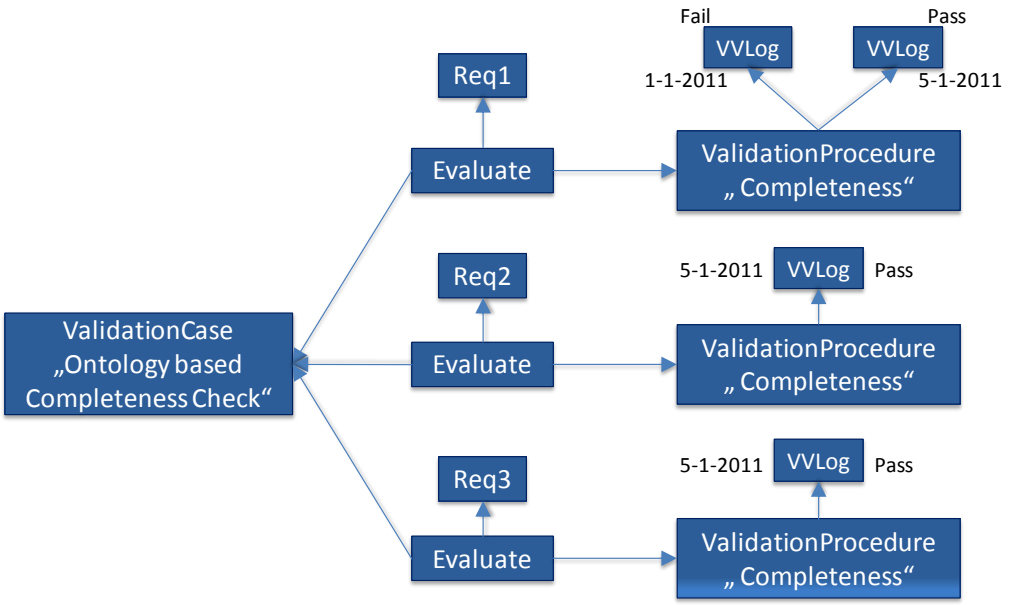

Abbildung 70: Repräsentation von Vollständigkeitsanalyse

Alle genannten Vollständigkeitsanalysen werden auf die gleiche Weise abgespeichert.

Verfolgbar

Hierfür werden keine weiteren V&V Daten benötigt. Alle in Abbildung 65 dargestellten Qualitätsmetriken lassen sich basierend auf der Anforderungsverfolgungsstruktur auswerten.

Prozessinstanziierung

Im nächsten Schritt sind die bisher produktorientierten Metriken durch eine Prozessinstanziierung in den jeweiligen spezifischen Kontext zu setzen. Die Instanziierung liefert die Kalibrierung und ermöglicht auch indirekt die Bewertung von Prozessanforderungen. Die Instanziierung der Prozesse für einen Beispieldatensatz demonstriert die Anwendung der prozessorientierten Qualitätsdefinition für die ISO26262 System Design Phase.

Kriterium bei der Prozessinstanziierung ist die ASIL Klassifikation der im System enthaltenen Komponenten. Abbildung 71 zeigt eine Instanziierung für Teile einer System Architektur eines ASIL C Systems mit zwei nach ASIL A und B klassifizierten Subkomponenten. Am Anfang steht die Entwicklung des technischen Sicherheitskonzepts. Hierzu gehören z.B. die Spezifikation und Validierung technischer Sicherheitsanforderungen sowie die Entwicklung der Sicherheitsarchitektur selbst. Ist eine erste Version der Architektur fertiggestellt, lassen sich weitere Abschnitte der System Design Phase instanziieren. So z.B. notwendige Anforderungsformalisierungs- und Design Verifikationsschritte entsprechend der resultierenden ASIL Klassen der Komponenten.

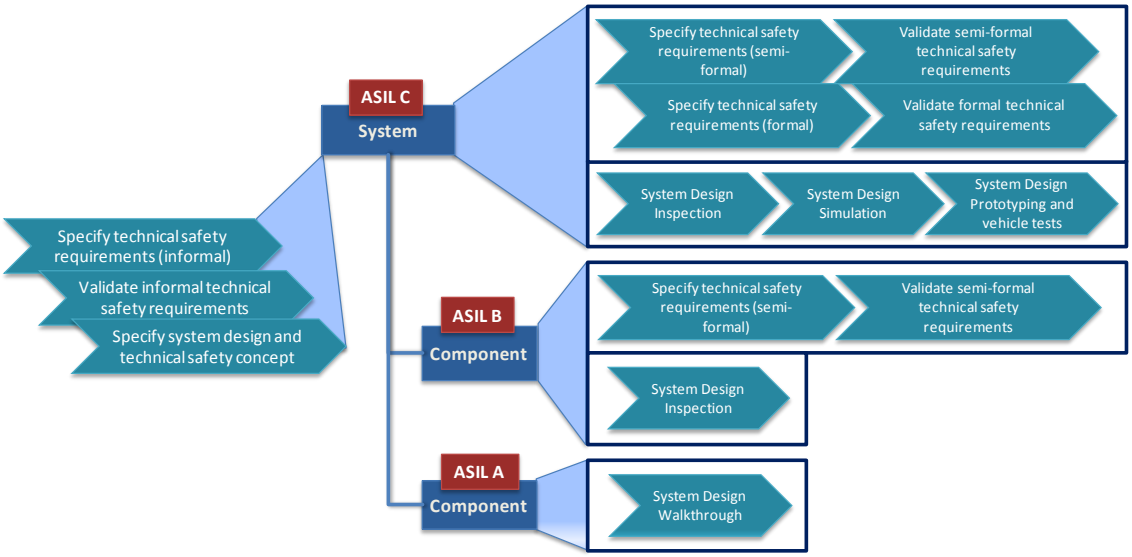

Abbildung 71: Prozessinstanziierung am Beispiel der Anforderungsdefinition und System Verifikation

Für jede Komponente entsteht entlang einer ohnehin durchzuführenden Prozesskalibrierung eine zugeschnittene Qualitätsdefinition, die den Prozesskontext berücksichtigt. So ist das Gesamtsystem entsprechend ISO26262 erst verifiziert, sobald sowohl „System Design Inspection“ als auch „System Design Simulation“ und „System Design Prototyping“ erfolgreich durchgeführt wurden und aus Sicht jeder Task die Anforderungen erfüllt sind. Der letzte Abschnitt beschreibt die Auswertung entsprechend der erzeugten projektspezifischen Qualitätsdefinition.

Qualitätsbewertung

Mittels MS Project lässt sich die dargestellte Prozessinstanz mit Start- und Endterminen, Ressourcen und weiteren Abhängigkeiten versehen. Definierte Anforderungen, Validierungs- und Verifikationsergebnisse werden regelmäßig mit dem Repository synchronisiert, so dass zu jedem Zeitpunkt der Entwicklungsstand pro Komponente und pro Prozessschritt basierend auf aktuellen Entwicklungsdaten (z.B. aus den bereits genannten Werkzeugen) auswertbar ist.

Abbildung 72 demonstriert anhand eines Ausschnitts der einheitlichen Modellsicht, wie Prozesskontext abhängig der Anforderungserfüllungsgrad aus Sicht der Tasks „Verifiziere System Design durch Simulation“ und „Verifiziere System Design durch Prototyping“ gemessen wird. Beide Tasks verifizieren die „Technischen Sicherheitsanforderungen“ (kurz TSA) und liefern Verifikationsergebnisse, jedoch mit jeweils unterschiedlichen Methoden. Dieser prozessspezifische Parameter ist durch die folgende Implementierung der Qualitätsmetriken berücksichtigt:

- **Base Measure – Anforderungserfüllung (V&V: Simulation):**
  self.requirementFulfillment(„Simulation“) → sum()
- **Base Measure – Anzahl Anforderungen:**
  self → size()
- **Derived Measure – Anforderungserfüllungsgrad (durch Simulation):**
  Anforderungserfüllungsgrad / Anzahl Anforderungen

Die Metrik der anderen Task ist analog implementiert. Lediglich der Parameter für die „requirementFulfillment“ Methode ist „Prototype“.

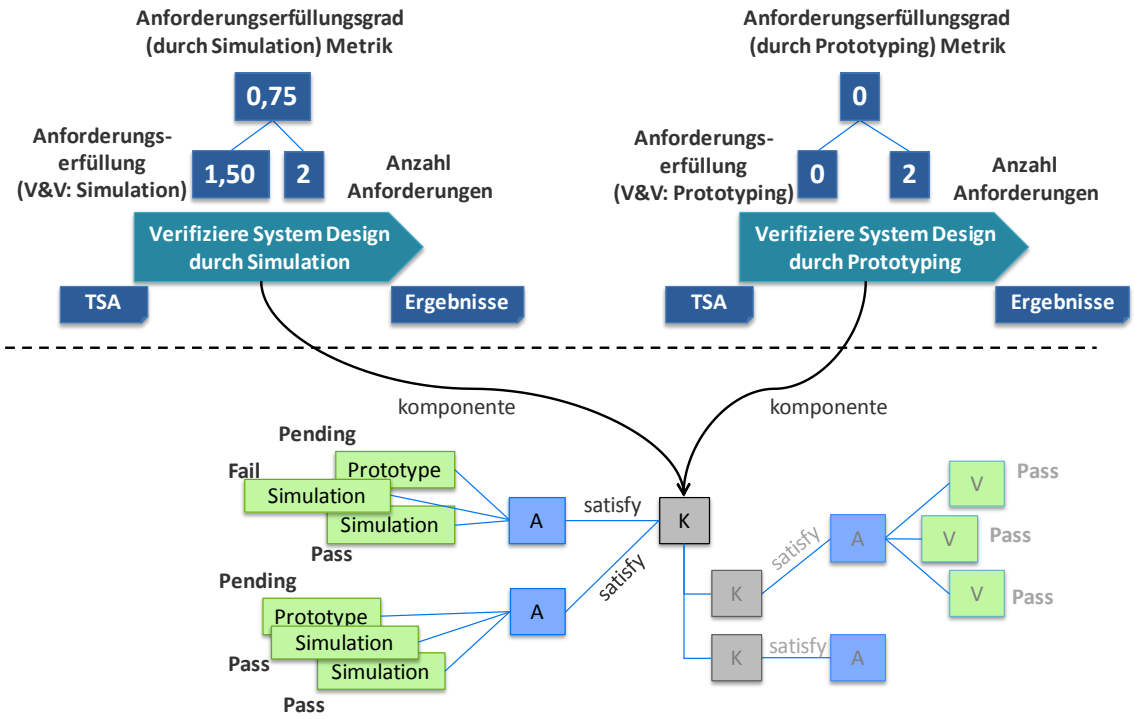

Abbildung 72: Qualitätsauswertung aus Sicht von zwei Verifikationstasks

Das Werkzeug evaluiert die Metriken entsprechend der OCL Abfrage der TSA gegen alle mit der Wurzelkomponente der Architektur verknüpften Anforderungen. Im Beispiel sind dies zwei. Die durch Simulation nachgewiesene Anforderungserfüllung beträgt 0,75 (75%), da drei von vier `VVCases` vom Typ „Simulation“ erfolgreich durchgeführt wurden. Die prototypische Erprobung steht jedoch noch aus. Alle Tests stehen auf „Pending“. Der Anforderungserfüllungsgrad beträgt 0%. Beide Qualitätsmetriken dienen als Indikator für ihre jeweilige Task.

5.2.4 Ergebnisse

Das Anwendungsbeispiel zeigt exemplarisch, wie eine systematische quantitative Qualitätsüberwachung für einen ISO26262 basierten Entwicklungsprozess umsetzbar ist. Die Ergebnisse lassen sich anhand der aufgestellten Ziele wie folgt zusammenfassen:

- **Einheitliche Qualitätsdefinition (Ziel 1):** Die prozessorientierte Qualitätsdefinition ermöglichte eine einheitliche Qualitätsdefinition für die ISO26262 System Design Phase. Ehemals getrennt betrachtete Qualitätscharakteristika wie Anforderungsqualität und Anforderungserfüllung sind zusammen mess- und interpretierbar. Das Ziel wurde **erfüllt**.
- **Kalibrierung auf einzelnes Projekt (Ziel 2):** Der Ansatz zur prozessorientierten Qualitätsdefinition stellt eine geeignete Grundlage für eine effiziente Kalibrierung auf einzelne Projekte dar. Entlang einer ASIL basierten Prozessinstanziierung, erfolgte die Kalibrierung der Qualitätsdefinition ohne zusätzlichen Aufwand. Das Ziel wurde **erfüllt**.
- **Integration V&V Ergebnisse (Ziel 3):** Die Zusammenführung von Analyseergebnissen verschiedener Werkzeuge zur Anforderungsdefinition und -validierung wurde demonstriert. Das Ziel wurde **erfüllt**.
- **Berücksichtigung von Prozesskontext (Ziel 6):** Der Ansatz zur prozessorientierten Qualitätsdefinition war Grundlage zur Berücksichtigung des Prozesskontexts. Die ASIL abhängig vorgeschriebenen Entwicklungsmethoden wurden bei der Qualitätsbewertung berücksichtigt. Die Umsetzung der Anforderungserfüllungsmetrik demonstriert dies. Je nach betrachteter Task, wird die verwendete Verifikationsmethode bei der Berechnung berücksichtigt. Das Ziel wurde **erfüllt**.

Die Evaluation verdeutlicht, dass sich die entwickelte Methodik für eine prozessorientierte Qualitätsbewertung, gerade für den Einsatz im sicherheitskritischen Umfeld sehr gut eignet. Dies lässt sich anhand der zentralen Eigenschaften des entwickelten Ansatzes begründen:

- **Anforderungsorientierung** –Anforderungsqualität und -erfüllung eignen sich als Qualitätsindikatoren insbesondere für das Management funktionaler Sicherheit nach ISO26262, da Anforderungen, deren Verfolgung, Validierung und Verifikation den Sicherheitsstandard prägen. Beide Konzepte ließen sich einfach

auf die ISO26262 Prozessschritte übertragen. Ferner ist für den entwickelten Ansatz eine sorgfältige Anforderungsverfolgung Voraussetzung. Dies ist im sicherheitskritischen Umfeld am ehesten gegeben, da die Verfolgung von Anforderungen entlang des Entwicklungsprozesses nachzuweisen ist.

- **Prozessorientierung** – Der ISO26262 Entwicklungsprozess unterscheidet sich je nach ASIL Kritikalität des betrachteten Systems und seiner Subkomponenten. Durch die prozessorientierte Definition der Qualitätsmetriken ist garantiert, dass bei einer Anwendung des ISO26262 Referenzprozesses in einem Projekt eine projektspezifische Qualitätsdefinition entsteht.

Abschließend ist anzumerken, dass die hier beschriebenen Prozesse und Qualitätsmetriken nur als ein mögliches Beispiel anzusehen sind und keinen Anspruch auf Vollständigkeit erheben. Unternehmen müssen die Interpretation des Standards (z.B. dass „++“ immer im Prozess enthalten sind und + nur optional) nicht teilen. Um den Ansatz auf einen unternehmensspezifischen Entwicklungsprozess anzuwenden, sind hierzu aber lediglich die Prozessbibliotheken anzupassen und die Interpretation der ISO26262 durch die hier nicht betrachteten Abschnitte zu vervollständigen.

5.3 Zusammenfassung

Das vorliegende Kapitel zur Evaluation hat sowohl am Beispiel eines konkreten AMS Entwicklungsprojekts, als auch mit Hilfe der Entwicklung eines Konzepts zur Überwachung eines Entwicklungsprozess gemäß ISO26262 gezeigt, dass der entwickelte Ansatz den aufgestellten Zielen dieser Arbeit genügt und sich für ein entwicklungsbegleitendes Qualitätsmonitoring eignet. Die Arbeit ist daher im Wesentlichen als erfolgreich zu bewerten. Hierzu trägt auch die Relevanz der betrachteten Prozesse sicherheitskritischen Elektronik und Analog Mixed Signal Entwicklung bei. Die folgende Tabelle gibt einen Überblick der Zielerfüllung, nachgewiesen durch jeweils eines oder durch beide Anwendungsbeispiele.

Ziel der Arbeit	AMS	ISO 26262	Anmerkung
Z1: Einheitliche Qualitätsdefinition	erfüllt	erfüllt	In beiden Beispielen wurde eine auf formalen Metamodellen basierte Qualitätsdefinition aufgestellt, die klar definierten Regeln folgt. Beide Anwendungsbeispiele demonstrieren, wie man mit den entwickelten Modellen verschiedene Qualitätskriterien messbar formalisiert.
Z2: Effiziente Kalibrierung auf einzelne Projekte	n/a	erfüllt	Das ISO26262 Beispiel demonstriert sowohl die Anforderungsorientierung als auch die Kalibrierung einer Qualitätsdefinition ausge-

			hend von einem definierten Prozessmodell auf ein konkretes Projekt. Eine projektspezifische Auswertung ist garantiert, die Wiederverwendung von Qualitätsdefinitionen aber auch nicht außer Acht gelassen
Z3: Integration V&V Ergebnisse	erfüllt	erfüllt	In beiden Anwendungsbeispielen wurde demonstriert, wie durch Verwendung des V&V Metamodells Analyseergebnisse integriert wurden.
Z4: Domänenübergreifende Qualitätsbetrachtung	teilweise	teilweise	Jedes Anwendungsbeispiel für sich erfüllt dieses Ziel nur teilweise. Beide betrachten jeweils nur einen bestimmten Teil der Entwicklung (Architektur bzw. Analog Design). In ihrer Kombination zeigt sich jedoch, dass das Qualitätsmodellierungsframework verschiedene Produktpartialmodelle abbilden kann. Eine domänenübergreifende Qualitätsbetrachtung ist prinzipiell möglich.
Z5: Entwicklungsbegleitende Auswertung	erfüllt	n/a	Wie am AMS Beispiel gezeigt, ist das Qualitätsmodellierungsframework prinzipiell in der Lage, eine entwicklungsbegleitende Auswertung durchzuführen. Datenerfassung und –analyse ist an beliebigen Zeitpunkten der Entwicklung initialisierbar.
Z6: Berücksichtigung von Prozess- und Projektkontext	erfüllt	erfüllt	Die Definition von Qualitätsmetriken entlang eines formalen Prozessmodells war der Schlüssel zur Erfüllung dieses Ziels. Instanziiert man die Prozesse wird je nach Task genau das gemessen, was gerade von Interesse ist. Besonders das ISO26262 Beispiel verdeutlicht dies. So wird Anforderungserfüllung immer nur entsprechend des Prozesstasks und der dort verwendeten Verifikationsmethode bewertet. Das AMS Beispiel demonstriert die Anbindung an die Projektsteuerung und berücksichtigt definierte Projektziele.
Z7: Automatische Datenerfassung, -integration und -	erfüllt	n/a	Das AMS Beispiel demonstrierte, wie mittels Transformationen, welche die proprietären Daten auf das entwickelte Produktmetamodell

analyse			abbilden, eine automatische Datenerfassung und –integration zu gewährleisten ist. Formalisierte Metriken erlauben die automatische Datenanalyse.
Z8: Anpassbarkeit an bestehende Entwicklungsprozesse	teil-weise	n/a	Die Anpassbarkeit erfolgt über Modelltransformationen aus den jeweils verwendeten Datenformaten in die definierten Metamodelle. Im AMS Beispiel wurde dies umgesetzt, im ISO26262 Beispiel konzeptionell beschrieben. Nicht belastbar bewertet wurde die Frage, ob Entwickler bei ihrer Arbeit unbeeinflusst sind, da keine Begleitung eines laufenden Projekts durchgeführt wurde.

Tabelle 19: Zielerfüllung der Arbeit

Lediglich die Erfüllung von Ziel 8 hinsichtlich der angestrebten Anpassbarkeit an bestehende Entwicklungsprozesse ohne die Entwickler bei ihrer Arbeit zu beeinflussen, ist nur teilweise beantwortet. Dies ist insgesamt nicht weiter verwunderlich, da der Nachweis der Erfüllung dieses Ziels die Begleitung eines gerade laufenden Entwicklungsprojekts erfordert. Um in der Praxis tatsächliche Serienentwicklung zu begleiten, bedarf es jedoch noch einer Verbesserung der prototypischen Implementierung, um dessen Reifegrad zu erhöhen.

Das folgende Kapitel zum Abschluss der Arbeit greift die Probleme und Grenzen der Arbeit noch einmal auf und wirft einen Blick auf mögliche Erweiterungen und Detaillierungen des Themas, sowie angrenzenden Arbeiten und Forschungsfragen für welche diese Arbeit Grundlage sein kann.

6 Zusammenfassung und Ausblick

Die Fähigkeit frühzeitig auf Qualitätsprobleme während der Entwicklung zu reagieren ist ein Schlüsselfaktor der heutigen komplexen Elektronikentwicklung. Dies ist ohne Zweifel anerkannt, trotzdem gelingt dies nicht immer oder eben nur mit mehr Aufwand als notwendig wäre. Gründe hierfür sind vor allem die hohe und stetig steigende Komplexität heutiger Elektronik und der Gesamtsysteme in die sie eingebettet ist, enge Markteintrittsfenster („time to market“) und die Integration verschiedener Fachrichtungen. Hinzu kommen monetäre und personelle Restriktionen, die Notwendigkeit zur Berücksichtigung aller Entwicklungsphasen sowie eine Vielzahl verwendeter Werkzeuge.

Die Arbeit adressiert dieses Problem in den vorangegangenen Kapiteln zum Stand der Technik, der Konzeption und Umsetzung eines entwicklungsbegleitenden prozessorientierten Qualitätsmonitoring und dessen Anwendung in zwei Anwendungsbeispielen aus der Elektronikentwicklung. Das abschließende Kapitel rundet die Darstellung mit einer Zusammenfassung der wissenschaftlichen Beiträge (6.1) und einem Ausblick auf zukünftige Erweiterungen (6.2) ab.

6.1 Zusammenfassung des Beitrags

Der Abgleich der entwickelten Lösung mit den definierten Zielen aus Abschnitt 1.4.1 ist bereits zum Abschluss des Evaluationskapitels in Abschnitt 5.3 erfolgt. Zum Abschluss dieser Arbeit geht diese noch einmal auf den Hauptbeitrag ein, der

„Erstellung eines Konzepts zur entwicklungsbegleitenden Produktqualitätsbewertung basierend auf einer prozessorientierten Qualitätsdefinition und einer einheitlichen Repräsentation qualitätsrelevanter Produktdaten unter Verwendung integrierter Produkt- und Prozessmetamodelle.“

Die Dissertation liefert hierfür folgende wissenschaftlichen Beiträge:

1) **Umfassende Untersuchung existierender Methoden und Modelle für die Qualitätsbewertung in der Produktentwicklung:** Dieser in Kapitel 2 gelieferte Beitrag führt zum einen in die drei Themenkontexte des Projektcontrollings, des Requirements Engineering und der Qualitätsmodellierung und -bewertung ein, in denen sich die vorliegende Arbeit bewegt. Zum anderen identifiziert das Kapitel Schwachstellen existierender Ansätze und leitet daraus den Handlungs-

bedarf ab. Dieser umfasst vor allem a) eine einheitliche Sicht auf qualitätsrelevante Daten b) einen systematischen Ansatz zur Kalibrierung auf einzelne Projekte und c) eine prozessorientierte Bewertung zur Unterstützung von Prozesskontrolle und –steuerung basierend auf aktuellen Entwicklungsständen. Der Handlungsbedarf dient während der Konzeption als Motivation für verschiedene Entwurfsentscheidungen.

2) **Konzeption eines modellbasierten Vorgehens für ein prozessorientiertes Qualitätsmonitoring als Grundlage für die entwicklungsbegleitende Kontrolle und Steuerung von Projekten:** Die zentrale Idee dieser Arbeit erfährt eine ausführliche Beschreibung in Kapitel 3. Es beschreibt einen Ansatz für ein prozessorientiertes Qualitätsmonitoring, welcher bei einer einheitlichen Qualitätsdefinition, deren Kalibrierung auf einzelne Projekte und ihrer entwicklungsbegleitenden Auswertung hilft. Die Definition von Qualitätsmetriken und -zielen entlang eines Prozessmodells ermöglicht eine effiziente Kalibrierung unternehmensweiter Qualitätsdefinitionen zusammen mit der Instanziierung eines Prozesses für ein konkretes Projekt. Gleichzeitig sichert die Definition von Prozess- und Metrik Bibliotheken eine mögliche Wiederverwendung. Das zu Grunde liegende Qualitätsmodellierungsframework verknüpft diese prozessorientierte Qualitätsdefinition auf effiziente Weise mit einer produktorientierten Sicht, die alle anfallenden Daten wie definierte Anforderungen und V&V Ergebnisse integriert.

3) **Ausführliche Evaluation der Lösung anhand zweier Beispiele aus der Elektronikentwicklung:** Unter Verwendung der in Kapitel 4 beschriebenen prototypischen Umsetzung des Konzepts befasst sich das fünfte Kapitel mit den dargelegten Zielen und ihrer Erfüllung durch die vorliegende Arbeit. Es verifiziert die Anwendbarkeit der Konzeption in Bezug auf die Problemstellung im Rahmen industrieller Entwicklungsprozesse. Im ersten Anwendungsbeispiel findet die Methodik in einem Analog Mixed Signal Entwicklungsprozess der Robert Bosch GmbH Anwendung. Alle notwendigen V&V Ergebnisse wurden integriert, um eine entwicklungsbegleitende Auswertung definierter Qualitätsmetriken zu unterstützen. Ferner wurde eine Operationalisierung der Qualitätsbewertungen für die Projektsteuerung durch die Anbindung einer bei Bosch entwickelten Projektplanoptimierungskomponente erreicht. Das zweite Anwendungsbeispiel evaluiert das Konzept im Kontext des Automotive Sicherheitsstandard ISO26262. Entlang der System Design Phase wurde eine prozessorientierte Qualitätsdefinition implementiert und speziell deren Kalibrierung auf einzelne Projekte demonstriert.

Neben einer Zusammenfassung der wissenschaftlichen Beiträge, stellt der letzte Abschnitt mögliche Erweiterungen und Anschlussmöglichkeiten an die vorliegende Arbeit dar.

6.2 Ausblick

Die vorliegende Arbeit nutzt die Qualitätsbewertung in erster Linie für das Projektcontrolling. Folgende weitere Anwendungen können von der entwickelten Lösung dieser Arbeit profitieren:

- **Einflussbewertung neuer Entwicklungsmethoden:** Arbeiten in diesem Umfeld streben eine Bewertung des Mehrwerts neuer Entwicklungsmethoden auf existierende Entwicklungsprozesse an (siehe z.B. (Koppe, 2010a), (Koppe, 2010b)). Verkürzt sich durch eine neue Methode die Entwicklungszeit? Erhöht sich meine Produktqualität? Grundlage solcher Bewertungen sind Kenntnisse über die Wirkzusammenhänge in der Produktentwicklung. Wie lange brauche ich z.B. mit einer Review basierten Verifikation, um bestimmte Qualitätsziele zu erreichen. Die in dieser Arbeit geschaffene Möglichkeit einer entwicklungsbegleitenden Qualitätsbewertung erlaubt eine entwicklungsbegleitende Überwachung von Entwicklungsprozessen und der Bildung solcher Wirkzusammenhangsmodelle. Eine Integration beider Ansätze verspricht eine einheitliche modellbasierte Methodik zur Adressierung der Prozessanalyse und –optimierung.
- **Workflow gesteuerte Entwurfsprozesse:** Die prozessorientierte Qualitätsdefinition erlaubt nicht nur deren Operationalisierung für das Projektmanagement. Die enge Verknüpfung mit der Prozessdefinition legt prinzipiell die Grundlage, die Qualitätsbewertungsergebnisse für Anwendungen zu nutzen, die deutlich näher am Entwickler sind, so z.B. für eine agile und dynamische Workflow basierte Steuerung von Entwicklungsprozessen. Dies bedeutet, dass eine Workflow basierte Leitung von Entwicklern nicht mir nur auf statischen Workflows basiert, sondern den aktuellen Qualitätsstand mit in die Auswahl erlaubter Workflows und nächster Schritte einbezieht. Das Qualitätsmonitoring dieser Arbeit wäre bei einer entsprechenden Erweiterung die Basis einer aktiven Prozesssteuerung.

Abbildungen

Tabellen

Literatur

(Albinet, 2008) Albinet, A.: *The MeMVaTEx methodology: from requirements to models in automotive application design*. In: 4th European Congress ERTS (Embedded Real Time Software), Toulouse, France 2008

(Alt, 2007) Alt, J., Badstübner, F., Brand, H.J., Jentzsch, E., Sebeke, C. und Vörg, A.: *PRODUKTIV+: Messung der Produktivität beim Entwurf nanoelektronischer Systeme - Ein Meßsystem für die Wirksamkeit von Electronic Design Automation*. edaCentrum Newsletter, 03/2007, 2007

(Adejouma, 2010) Adedjouma, M., Dubois, H., Maaziz, K. und Terrier, F.: *A Model-Driven Requirement Engineering Process Compliant with Automotive Domain Standards*. In Proceedings of MDTPI 2010. Paris, France, 2010.

(Aldazabal, 2008) Aldazabal, A ., Baily, T., Nanclares, F., Sadovykh, A., Hein, C., Esser M. und Ritter, T.: *Automated Model Driven Development Processes*. In Proceedings of the ECMDA workshop on Model Driven Tool and Process Integration, Fraunhofer IRB Verlag, Stuttgart, 2008

(Alur, 1994) Alur, R. und Dill, D.: *A theory of timed automata; Theoretical Computer Science*, 126(2): S. 183–235, 1994

(Arden, 2010) Arden, W., Brillouët, M., Cogez, P., Graef, M., Huizing, B. und Mahnkopf, R.: *More than More*. White Paper, 2010

(Armengaud, 2011) Armengaud, E., Zoier, M., Baumgart, A., Biehl, M., Chen, D., Griessnig, G., Hein, C., Ritter, T. und Kolagari, R. T.: *Model-based toolchain for the efficient development of safety-relevant automotive embedded systems*. SAE 2011 World Congress & Exhibition, 2011

(ARP4754, 1996) *Certification considerations for highly integrated or complex aircraft systems*. EUROCAE ED-79 and SAE Aerospace Recommended Practice ARP 4754,1996

(ATESST, 2008) Deliverable of the ATESST project, contract number 2004-026976. *EAST ADL 2.0 Specification*, 2008

(ATESST, 2010) Deliverable of the ATESST2 project, Grant Agreement 224442, *EAST ADL Domain Model Specification*, Version 2.1, 2010

(Bacchini, 2007) Bacchini, F., Hu, A.J., Fitzpatrick, T., Ranjan, R., Lacey, D., Tan, M., Piziali, A. und Ziv, A.: *Verification coverage: when is enough enough*. In Proceedings of the 44th annual conference on Design automation, San Diego, California, 2007

(Basili, 1988) Basili, V.R. und Rombach, H.D.: *The TAME project: towards improvement-oriented software environments*, In IEEE Transactions on Software Engineering, 14(6): S. 758-773, 1988

(Basili, 1994) Basili, V., Caldiera, G. und Rombach, H.D.: *The Goal Question Metric Approach*. In Encyclopedia of Software Engineering, John Wiley & Sons, Inc., S. 528-532, 1994

(Basili, 2002) Briand, L.C., Morasca, S. und Basili, V.R.: *An Operational Process for Goal-Driven Definition of Measures*, In IEEE Transactions on Software Engineering, 28(12): S. 1106-1125, 2002.

(Berson, 1997) Berson, A. und Smith, S.J.: Data Warehousing, Data Mining, and OLAP. New York: McGraw-Hill, 1997

(Birnbaum, 2001) Birnbaum, M. und Johnson, C.C.: *VSIA quality metrics for IP and SoC*. In Proceedings of International Symposium on Quality Electronic Design, S. 279 – 283, 2001

(Blaschke, 2009a) Blaschke, J., Sebeke, C. und Rosenstiel, W.: *ASIC Design Project Management Supported by Multi Agent Simulation*. In Proceedings of the 5TH IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI'2009), S. 87-93, Thessaloniki, Greece. Springer 2009

(Blaschke, 2009b) Blaschke, J., Sebeke, C. und Rosenstiel, W.: U*sing genetic algorithms for planning of ASIC chip-design project flows*. In Proceedings of the Eleventh conference on Congress on Evolutionary Computation (CEC'09), S. 1881-1888, 2009

(Blaschke, 2010) Blaschke, J., Sebeke, C. und Rosenstiel, W.: *Organizing and Planning the ASIC Design Process by Means of a Multi-agent System*. In Proceedings of the International Conference on Agents and Artificial Intelligence, Volume 1 (1): S. 459-463, Valencia, Spain, 2010

(Bøegh, 1999) Bøegh, J. Depanfilis, S., Kitchenham B. und Pasquini, A.: *A Method for Software Quality Planning, Control, and Evaluation*. In IEEE Software. 16(2): S. 69–77, 1999

(Boehm, 1976) Boehm, B. W., Brown, J.R. und Lipow, M.: *Quantitative evaluation of software quality*. In Proceedings of the 2nd International Conference on Software Engineering (ICSE), S. 592-605, 1976

(Boehm, 1978) Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., MacLeod, G.J. und Merritt, M.J. *Characteristics of Software Quality*. North Holland Publishing Company, 1978

(Bowen, 1976) Bowen, T.P.: *Specification of Software Quality Attributes*. Rome Laboratory, New York, Technical Report, RADC-TR-85-37, Vols. 1-3, 1976

(Botella, 2001) Botella, P., Burgués, X., Franch, X., Huerta, M. und Salazar, G.: *Modeling non-functional requirements*. In Proceedings of Jornadas de Ingenieria de Requisitos Aplicada (JIRA), 2001

(Brand, 2004) Brand, H.J., Rülke, S. und Radetzki, M.: *IP Qualification for Efficient System Design*. In Proceedings of the 5th International Symposium on Quality Electronic Design (ISQED), 2004.

(Brcina, 2008) Brcina, R. und Riebisch, M.: *Defining Traceability Links Semantics for Design Decision Support*. Traceability Workshop at ECMDA, Berlin, Germany, 2008

(Broy, 2005) Broy, M., Deißenböck, F. und Pizka, M.: *A Holistic Approach to Software Quality at Work*. In Proceedings of the 3rd World Congress for Software Quality, 2005

(Broy, 2006) Broy, M., Deißenböck, F. und Pizka, M.: *Demystifying Maintainability.* In Proceedings of the 4th Workshop on Software Quality, S. 21-26 2006

(Buchel, 1987) Büchel, A.: P*rojektmanagement*. Eidgenössische Technische Hochschule (ETH), Zürich, 1987

(Burghardt, 2002) Burghardt, M.: *Projektmanagement – Leitfaden für die Planung, Überwachung und Steuerung von Entwicklungsprojekten*. 6.Auflage, Publicis MCD-Verlag, Erlangen, 2002

(Buschermoehle, 2006) Buschermöhle, R., Eekhoff, H. und Josko, B.: *SUCCESS Rate and Factors of IT - Projects 2006 in Germany*. Software Engineering Research and Practice, S. 644-650, 2007

(Cancila, 2009) Cancila, D., Terrier, F., Belmonte, F., Dubois, H., Espinoza, H. und Gérard, S.: *Model-Based Safety Engineering.* In: 2nd International Workshop on Model Based Architecting and Construction of Embedded Systems (ACES-MB 2009) in conjunction with MODELS Conference, Denver, Colorado, USA (2009)

(CESAR, 2010a) CESAR Project, *Requirement Description for automotive domain – V2*, Technical Report, 2010

(CESAR, 2010b) CESAR Project, *Requirement Description for Aerospace domain – V2*, Technical Report, 2010

(CESAR, 2010c) CESAR Project, *Requirement Description for Automation and Railway domain – V2*, Technical Report, 2010

(CESAR, 2010d) CESAR Project, *RE Language Definitions to formalize multi criteria requirements – V2*, Technical Report, 2010

(CESAR, 2010e) CESAR Project, *Detailed Analysis Report on Requirements Engineering*, Technical Report, 2010

(Ciolkowski, 2007) Ciolkowski, M., Heidrich, J., Münch, J., Simon, F. und Radicke, M.: *Evaluating Software Project Control Centers in Industrial Environments*. In Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM), S.314-323, Madrid, Spain. 2007

(Clarke, 1994) Clarke, E., Grumberg, O. und Long, D.: *Model Checking and Abstraction*. *TOPLAS*, 16(86): S. 1512-1542, 1994

(Clarke, 1999) Clarke, E., Grumberg, O. und Peled, D.: *Model Checking*. MIT Press, 1999.

(Cleland-Huang, 2003) Cleland-Huang, J., Chang, C.K. und Wise J.: *Automating Performance Related Impact Analysis through Event Based Traceability*. In Requirements Engineering Journal, 8(3): S. 171-182, Springer-Verlag, 2003

(Cleland-Huang, 2002) Cleland-Huang, J., Chang, C.K., Hu, H., Javvaji, K., Sethi, G. und Xia, J.: *Requirements Driven Impact Analysis of System Performance*. In IEEE Proceedings of the Joint Conference on Requirements Engineering, Essen, 2002

(CMMI, 2010) Carnegie Mellon University Software Engineering Institute, *CMMI for Development*, Version 1.3., 2010

(Commes, 1983) Commes, M. Th. und Lienert, R.: *Controlling im FuE Bereich*, Zeitschrift Führung und Organisation, 52 Jg, S. 347-354, 1983

(Crosby, 1980) Crosby, P.B.: *Quality is free*, Reissue Edition, Signet, 1980

(Crosby, 1995) Crosby, P. B.: Q*uality Without Tears: The Art of Hassle-Free Management*. McGraw-Hill, 1995.

(Damm, 2005) Damm, W.: *Controlling Speculative Design Processes Using Rich Component Models*. In Proceedings of the Fifth International Conference on Application of Concurrency to System Design, S. 118-119, 2005

(Damm, 2009a) Damm, W., Achatz, R., Beetz, K., Broy, M., Daembkes, H., Grimm, K. und Liggesmeyer, P.: *Nationale Roadmap Embedded Systems*. 2010

(Damm, 2009b) Damm, W.: Contract-Based Analysis of Automotive and Avionics Applications: The SPEEDS Approach. In Proceedings of Formal Methods of Industrial for Industrial Critical Systems, Springer, S. 3, 2009

(Davis, 1993) Davis, A.: *Software Requirements: Objects, Functions and States*. Prentice Hall. 1993

(de Lucia, 2009) De Lucia, A., Oliveto, R. und Tortora, G. *Assessing IR-based Traceability Recovery Tools through Controlled Experiments*, In Empirical Software Engineering, 14(1): S. 57-92, 2009

(Deißenböck, 2007) Deißenböck, F., Wagner, S., Pizka, M., Teuchert, S. und Girard, J.F. *An Activity-Based Quality Model for Maintainability*. In Proceedings of the 23rd IEEE International Conference on Software Maintenance (ICSM '07), S. 184-193, 2007

(Deißenböck, 2008) Deißenböck, F., Juergens, E., Hummel, B., Wagner, S., Mas y Parareda, B. und Pizka, M.: *Tool Support for Continuous Quality Control*. In *IEEE Software*, 25(5): S. 60-67, 2008

(Deißenböck, 2009) Deißenböck, F.: *Continuous Quality Control of Long-Lived Software Systems*, Dissertationsarbeit, Technische Universität München, 2009

(Deming, 2000) Deming, W.E.: *Out of the Crisis*. MIT Press, 2000.

(Deutsch, 1988) Deutsch, M. und Willis, R.: *Software Quality Engineering*. Prentice-Hall, 1998

(DO178B, 1992) *Software considerations in airborne systems and equipment certification*, EUROCAE ED-12 and RTCA DO-178, issue B, 1/12/1992

(Dörr, 2003) Dörr, J.; Kerkow, D., Knethen, A. und von, Paech, B.: *Eliciting efficiency requirements with use cases*. 9th International Workshop on Requirements Engineering. Foundation for Software Quality, REFSQ'03. Pre-Proceedings, S.23-32, 2003

(Dörr, 2005) Dörr, J., Kerkow, D., Koenig, T., Olsson, T. und Suzuki, T.: *Non-Functional Requirements in Industry – Three Case Studies Adopting an Experience based NFR Method*. In Proceedings of the 13th International Conference on Requirements Engineering (RE'05), S 373–382. IEEE Computer Society, 2005

(Dromey, 1995) Dromey, R.G.: *A model for software product quality*. In IEEE Transactions on Software Engineering, 21(2): S. 146-162, 1995

(Dromey, 1996) Dromey, R.: *Cornering the Chimera*. In IEEE Software, 13(1): S. 33-43, 1996

(ECSS-E-ST-10C, 2009) European Cooperation for Space Standardization: *Space Engineering – System engineering general requirements*. ECSS-E-ST-10C, 2009

(ECSS-E-ST-40C, 2009) European Cooperation for Space Standardization: *Space Engineering – Software*. ECSS-E-ST-40C, 2009

(ECSS-Q-ST-80C, 2009) European Cooperation for Space Standardization: *Space Engineering – Software product assurance*. ECSS-E-ST-80C, 2009

(Egyed, 2003) Egyed, A.: *A Scenario-Driven Approach to Trace Dependency Analysis*, In IEEE Transactions on Software Engineering, 9(2): S. 116-132, 2003

(Egyed, 2005a) Egyed, A., Biffl, S., Heindl, M. und Grünbacher, P.: *A Value-Based Approach for Understanding Cost-Benefit Trade-Offs During Automated Software Traceability*. In Proceedings of the 3rd ACM International Workshop on Traceability in Emerging Forms of Software Engineering, S.2-7, ACM Press, 2005

(Egyed, 2005b) Egyed, A.: *Tailoring Software Traceability to Value-based Needs*. In: Biffl, S., Aurum, A., Boehm, B.W., Erdogmus, H., und Grünbacher, P. (eds.), *Value-Based Software Engineering*, Springer, 2005

(Ehrlenspiel, 2007) Ehrlenspiel, K.: *Integrierte Produktentwicklung*, 3. Auflage, Hanser, 2007

(Eigner, 2009) Eigner, M. und Stelzer, R.: *Product Lifecycle Management: Ein Leitfaden für Product Development und Life Cycle Management*. 2.Auflage, Springer Verlag, 2009

(Fagan, 1976) Fagan, M.E.: Design and code inspections to reduce errors in program development. IBM Systems Journal, 15(3): S.182-211, 1976

(Farfeleder, 2011) Farfeleder, S., Moser, T., Krall, A., Stalhane, T., Zojer, H. und Panis, C.: *DODT: Increasing requirements formalism using domain ontologies for improved embedded systems development*. In Proceedings of 14th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), S. 271 – 274, 2011

(Farrahi, 2000) Farrahi, A.H., Hathaway, D. I., Wang, M. und Sarrafiadeh, M.: *Quality of EDA CAD Tools: Definitions, Metrics and Directions*. In Proceedings of the IEEE International Symposium On Quality Electronic Design, S. 395-405, 2000

(Fast, 2005) Fast GmbH, Technische Universität München, *Study of Worldwide Trends and R&D Programmes in Embedded Systems in View of Maximising the Impact of a Technology Platform in the Area*, 2005, Zugriff: ftp://ftp.cordis.europa.eu/pub/ist/docs/embedded/final-study-181105_en.pdf

(Feldmuller, 2005) Feldmüller, D., Frick, A. und Grau, N.: *Welche Kompetenzen benötigt das IT-Projektmanagement*. Arbeitsgruppe Projektmanagement der Gesellschaft für Informatik, 2005

(Fitting, 1996) Fitting, M.: *First-Order Logic and Automated Theorem Proving*, 2.Auflage. Springer, 1996

(ForTISS, 2011) ForTISS GmbH, *Mehr Software (im) Wagen: Informations- und Kommunikationstechnik (IKT) als Motor der Elektromobilität der Zukunft*, Abschlussbericht des vom Bundesministerium für Wirtschaft und Technologie geförderten Verbundvorhabens ForTISS-IKT-Systemarchitektur für Elektromobilität", 2011

(Franch, 1998) Franch, X.: *Systematic Formulation of Non-Functional Characteristics of Software*. In Proceedings of the 3rd International Conference on Requirements Engineering (ICRE'98), S. 174-181, 1998

(Fraunhofer, 2011) Fraunhofer ESK: *Garantierte Zuverlässigkeit für Softwarekomponenten*, Projekt CHESS, 2011, Zugriff: http://www.esk.fraunhofer.de/content/dam/esk/de/documents/PDB_Chess_dt.pdf

(Garcia, 2009) García, F., Ruiz, F., Calero, C., Bertoa, M.F., Valecillo, A., Mora, B. und Piattini, M.: *On the Effective Use of Ontologies in Software Measurement*. Knowledge Engineering Review Journal, 24(1): S. 23-40, 2009

(Garvin, 1984) Garvin, D. A.: *What does product quality really mean?* MIT Sloan Management Review, 26(1):S. 25–43, 1984

(Geiger, 2001) Geiger. W.: Qualität als Fachbegriff des Qualitätsmanagements. In: Zollondz (Hrsg.), S.801-810, 2001

(Gilmore, 1974) Gilmore, H.L.: *Product Conformance Cost*, Quality Progress, S. 16-19, 1974

(Glinz, 2005) Glinz, M.: *Rethinking the Notion of Non-Functional Requirements*. In Proceedings of the Third World Congress for Software Quality (3WCSQ 2005), Munich, Germany, S. 55-64, 2005

(Glinz, 2007) Glinz, M.: *On Non-Functional Requirements*, In Proceedings of the 15th IEEE International Requirements Engineering Conference (RE), S. 21-26, 2007

(Gonzales-Perez, 2005) Gonzalez-Perez und Henderson-Sellers: *Templates and Resources in Software Development Methodologies*. In Journal of Object Technology, 4(4): S. 173-190, 2005, Zugriff: http://www.jot.fm/issues/issue_2005_05/article5

(GPM, 2004) P.C.G.D. GPM Deutsche Gesellschaft für Projektmanagement e.V., *Studie zur Effizienz von Projekten in Unternehmen*, 2004

(Göknil, 2008) Göknil, A., Kurtev, I. und van den Berg, K.G.: *A Metamodeling Approach for Reasoning about Requirements.* In: 4th European Conference Model Driven Architecture - Foundations and Applications, ECMDA-FA, Berlin, Germany, 2008

(Gotel, 1994) Gotel, O. and Finkelstein, A.W.: *An analysis of the requirements traceability problem*. In Proceedings of the International. Conference Requirements Engineering, IEEE Computer Society Press, S. 94–102, Colorado Springs, 1994

(Grady, 1987) Grady, R.B., Caswell, D.L.: *Software Metrics: Establishing a Company-Wide Program*, Prentice Hall, Upper Saddle River, NJ, 1987

(Hall, 2009) Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. und Witten, I.H.: *The WEKA Data Mining Software: An Update*, SIGKDD Explorations, 11(1): S.10-18, 2009

(Harrington, 1991) Harrington, H.J.: *Bussiness Process Improvement: The Breakthrough Strategy for Quality, Productivity, Competitiveness*", McGraw-Hill Professional, 1991

(Harmuth, 2003) Harmuth, U.: *Erfolgsfaktoren für Projekte. Analyse von PM-Award-Projekten nach gemeinsamen Erfolgsfaktoren*, 2003.

(Hausmann, 2009) Hausmann, K.: *Permeter: Performanzmessung in der Produktentwicklung auf Basis semantisch integrierter Produktmodelle*, Dissertationsarbeit, Universität Oldenburg, 2009

(Haumer, 2005) Haumer, P.: IBM Rational Method Composer: Part 1 Key concepts, 2005, http://www.ibm.com/developerworks/rational/library/dec05/haumer/

(Häusler, 2007) Häusler, S., Poppen, F., Hausmann, K., Preis, S., Hahn, A., Nebel, W., Leppelt, P., Hassine, A. und Barke, E.: *Modellierung von Komplexität und Qualität als Faktoren von Produktivität in Design-Flows für integrierte Schaltungen*, edaWorkshop07, VDE Verlag, 2007

(Häusler, 2009) Häusler, S., Sebeke, C., Blaschke, J., Rosenstiel, W. und Hahn, A.: Project Control System to Track and Optimize Chip Design Projects. In Proceedings of World Congress on Engineering and Computer Science 09, 2009

(Häusler, 2010) Häusler, S, Sebeke, C., Blaschke, J., Rosenstiel, W. und Hahn, A.: Chip Design Process Optimization Based on Design Quality Assessment. In IAENG Transactions on Engineering Technologies Volume 4: Special Edition of the World Congress on Engineering and Computer Science, S. 428-442, American Institute of Physics, 2010

(Heckerman, 1996) Heckerman, D.: *Bayesian networks for knowledge discovery*. In U.M. Fayyad, G. Piatetsky-Shapiro, P.Smyth and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*, S. 273-305, Cambridge, MA: MIT Press, 1996

(Heidrich, 2003) Heidrich, J.: *Custom-made Visualization for Software Project Control*. Technical Report 06/2003, Sonderforschungsbereich 501, University of Kaiserslautern, 2003

(Heidrich, 2006) Heidrich, J., Münch, J. und Wickenkamp, A.: *Usage Scenarios for Measurement-based Project Control*. In Proceedings of the 3rd Software Measurement European Forum (Smef 2006), Rom, Italien, 2006

(Hein, 2009) Hein, C., Ritter, T. und Wagner, M.: *Model-Driven Tool Integration with ModelBus*. Workshop Future Trends of Model-Driven Development, 2009

(Heindl, 2005) Heindl, M. und Biffl, S.: *A Case Study on Value-Based Requirements Tracing*, In Proceedings of the 10th European Software Engineering Conference, S. 60 – 69, 2005

(Heitmeyer, 2005) Heitmeyer, C., Archer, M., Bharadwaj, R. und Jeffords, R.: *Tools for constructing requirements specifications: The SCR toolset at the age of ten*, Computer System Science and Engineering Journal, vol.1: S. 19-35, 2005

(Horvath, 2006) Horváth, P.: *Controlling*, 10. Auflage, Vahlen, 2006.

(Hull, 2004) Hull, E., Jackson, K. und Dick, J.: *Requirements Engineering*, 2. Auflage, Springer Verlag, 2004

(Hyatt, 1996) Hyatt L. und Rosenberg L.: *A software quality model and metrics for identifying project risks and assessing software quality*, In Proceedings of 8th Annual Software Technology Conference, Utah, 1996

(IEEE, 1990) IEEE (1990). *Standard Glossary of Software Engineering Terminology*. IEEE Standard 610.12-1990, 1990

(IEEE, 1998) IEEE (1998). *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Std. 830-1998, 1998

(INCOSE, 1998) INCOSE Measurement Working Group (MWG): *Systems Engineering Measurement Primer*, 1998

(ISO9126-1, 2001) ISO/IEC: *ISO/IEC 9126-1 Software engineering- Product quality-Part 1: Quality model*. 2001

(ISO9126-2, 2002) ISO/IEC: *ISO/IEC 9126-3 Software engineering -Product quality-part3: Internal metrics*. 2002

(ISO9126-3, 2002) ISO/IEC: *ISO/IEC 9126-2 Software engineering -Product quality-part2: External metrics.* 2002

(ISO9126-4, 2002) ISO/IEC: *ISO/IEC 9126-4 Software engineering -Product quality-part4: Quality In Use metrics*. 2002

(ISO14598, 1999) ISO/IEC: *ISO/IEC 14598: Information Technology – Software Product Evaluation*, 1999

(ISO15939, 2007) ISO/IEC: *ISO/IEC 15939: Systems and software engineering — Measurement process*, 2007

(ISO25000, 2005) ISO/IEC: *ISO/IEC 25000:2005, Software Engineering—Software Product Quality Requirements and Evaluation (SQuaRE)—Guide to SQuaRE*, 2005

(ISO26262, 2010) International Organization for Standardization (ISO): *ISO/DIS 26262: Road Vehicles – Functional Safety*, Final Draft International Standard, 2010

(ISOWD29148.3, 2009) ISO/IEC: *ISO/IEC WD 29148.3: Software and systems engineering — Life cycle processes — Requirements engineering*, 2009

(Jost, 2010) Jost, H., Hahn, A., Häusler, S., Köhler, S., Gačnik, J., Köster, F. und Lemmer, K.: *Supportong qualification: Safety standard compliant process planning and monitoring*. In IEEE Symposium on Product Compliance Engineering (ISPCE), S.1-6, 2010

(Juran, 1988) Juran, J.M. und Defeo, J.: *Quality Handbook – The Complete Guide to Performance Excellence*, 6. Auflage, McGraw-Hill, 2010

(Katasonov, 2006) Katasonov A. und Sakkinen M.: *Requirements Quality Control: A Unifying Framework*. Requirements Engineering 11(1): S. 42-57, Springer-Verlag London, UK, 2006

(Kitchenham, 1987) Kitchenham, B.A.: *Towards a Constructive Quality Model*, Software Engineering Journal, S. 105-112, 1987

(Kitchenham, 1997) Kitchenham, B.A., Linkman, S.G., Pasquini, A. und V. Nanni: *The SQUID approach to defining a quality model*. Journal of Software Quality, vol. 6: S. 211-233, 1997

(Kitchenham, 2005) Al-Kilidar, H., Cox, K. und Kitchenham, B.A.: *The use and usefulness of the ISO/IEC 9126 quality standard*. In Proceedings of ISESE 2005. S. 126-132, 2005

(Koppe, 2010a) Koppe, R., Häusler, S., Poppen, F. und Hahn, A.: *Process Model Based Methodology for Impact Analysis of New Design Methods*. In Modelling and Management of Engineering Processes, S.53-64. Springer 2010

(Koppe, 2010b) Koppe, R., Häusler, S. und Hahn, A.: *Ein Modell zur Einflussanalyse von Änderungen in Entwicklungsprozessen*. INFORMATIK 2010, S. 639-644, 2010

(Kotonya, 1998) Kotonya, G. und Sommerville, I.: *Requirements Engineering: Processes and Techniques*. John Wiley & Sons, 1998

(Krammer. 2010) Krammer, M., Armengaud, E. und Bourrouilh, Q.: *Method Library Framework for Safety Standard Compliant Process Tailoring*. In: 37th EUROMICRO Conference on Software Engineering and Advanced Applications, 2011

(Lamsweerde, 2001) van Lamsweerde, A.: *Goal-Oriented Requirements Engineering: A Guided Tour*. In Proceedings of the 5th International Symposium on Requirements Engineering (RE'01), S. 249-261, Toronto, Canada, 2001

(Litke, 1995) Litke, H.-D.: *Projektmanagement*. Hanser Verlag, München, 1995.

(Mandl-Striegnitz, 1999) Mandl-Striegnitz, P. und Lichter, H.: *Defizite im Software-Projektmanagement - Erfahrungen aus einer industriellen Studie*. Informatik/Informatique, vol. 5, S. 4-9, 1999

(Martin, 1996) Martin A.E. und Shafer L.H.: *Providing a Framework for Effective Software Quality Assessment - A First Step In Automating Assessments*, Proceedings of the first Annual Software Engineering & Economics Conference, McLean, 1996

(Mastretti, 1995) Mastretti, M., Busi, M.L., Sarvello, R., Sturlesi, M. und Tomasello, S.: *VHDL Quality: Synthesizability, Complexity and Efficiency Evaluation*. In Proceedings of the conference on European design automation, S. 482-487, Brighton, England, 1995

(Mastretti, 1996) Mastretti M., Sturlesi M. und Tomasello S.: *Quality Measures and Analysis: A Way to improve VHDL Models*, Hardware Component Modeling, Kluwer Academic Publisher, 1996.

(Mayer, 1995) Mayer, R., Menzel, C., Painter, M., de Witte, P., Blinn, T. und Perakath, B.: *Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report*, Technical Report, 1995

(McCall, 1977) McCall, J. und Walters, G.: *Factors in Software Quality*. The National Technical Information Service, Springfield, VA, USA, 1977

(Münch, 2004) Münch, J., Heidrich, J.: *Software Project Control Centers: Concepts and Approaches*. Journal of Systems and Software, 70 (1): S. 3-19, Elsevier, 2004

(Nance, 1992) Nance, R.: *Software Quality Indicators: An Holistic Approach to Measurement*. In Proceedings of the 4th Annual Software Quality Workshop, Alexandria Bay, New York, 1992.

(OASIS, 2007) OASIS, *Web Services Business Process Execution Language Version 2.0*, 2007. Zugriff: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

(OMG, 2008) Object Management Group: *Software & Systems Process Engineering Meta-Model Specification Version 2.0*, 2008. Zugriff: http://www.omg.org /spec/SPEM/2.0

(OMG, 2010) OMG: *Object Constraint Language. Version 2.2*, 2010. Zugriff: http://www.omg.org/spec/OCL/2.2/

(OMG, 2011) Object Management Group (2011): *Business Process Model and Notation (BPMN), Version 2.0*, 2011. Zugriff: http://www.omg.org/spec/BPMN/2.0/

(Paquin, 2000) Paquin, J.P., Couillard, J. und Ferrand, D.J.: *Assessing and Controlling the Quality of a Project End Product: The Earned Quality Method*. In IEEE Transactions on Engineering Management, 47(1): S.88-89, 2000

(Park, 1996) Park, R.E., Goethert, W.B. und Florac, W.A.: *Goal-driven Software Measurement — A Guidebook*. Technical Report CMU/SEI-96-HB-002. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1996

(Pechlaner, 2005) Pechlaner, H., Tschurtschenthaler, P. und Peters, M.: *Erfolg durch Innovation*. Deutscher Universitäts-Verlag, 2005.

(Plösch, 2007) Plösch, R., Gruber, H., Hentschel, A., Körner, C., Pomberger, G., Schiffer, S., Saft, M. und Storck, S.: *The EMISQ Method - Expert Based Evaluation of Internal Software Quality*, In Proceedings of 3rd IEEE Systems and Software Week, Baltimore, USA, IEEE Computer Society Press, 2007

(Plösch, 2008) Plösch, R., Gruber, H., Pomberger, G., Schiffer, S. und Saft, M.: *Tool Support for Expert-Centred Code Assessments*, In Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation (ICST 2008, Lillehammer, Norwegen, IEEE Computer Society Press, 2008

(PMI, 2008) Project Management Institute: *A Guide to the Project Management Body of Knowledge: PMBoK Guide.* Fourth Edition PMI, 2008

(Projektmagazin, 2011) Glossar Projektmagazin, http://www.projektmagazin.de/glossarterm/projektcontrolling, Letzter Zugriff 3.6.2011

(Protheroe, 2000) Protheroe, D. und Pessolano, F.: *An objective measure of digital system design quality*. In Proceedings of the 1st International Symposium on Quality of Electronic Design, S.227, 2000

(Ramesh, 2001) Ramesh, B. und Jarke, M.: *Toward reference models for requirements traceability*, In IEEE Transactions on Software Engineering, S. 58-93, 2001

(Ross, 1977) Ross, D.T.: Structured Analysis (SA): A Language for Communicating Ideas. IEEE Transactions on Software Engineering, vol. 3(1), S. 16-34, 1977

(Rothlauf, 2010) Rothlauf, J.: Total Quality Management in Theorie und Praxis, 3.Auflage, Oldenbourg Verlag, 2010

(Scheer, 1992) Scheer, A.-W., Nüttgens, M. und Keller G.: *Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)"*. In Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Universität des Saarlandes, Heft 89, 1992

(Scheible, 2011) Scheible, J. und Pohlheim, H.: *Automated Model Quality Rating of Embedded Systems*. Im Tagungsband des 4. Workshop zur Qualitätsmodellierung und –bewertung (SQMB), S.25-34, 2011

(Sieg, 2007) Sieg, O.C.: *Ein Beitrag zur integrativen Unterstützung des Produktentwicklungscontrollings*. Maschinenbauinformatik 1/2007, Shaker Verlag, Aachen, 2007

(Sommerville, 2004) Sommerville, I.: *Software Engineering*, 7. Auflage. Addison-Wesley, 2004

(Spanoudakis, 2002) Spanoudakis, G.: *Plausible and adaptive requirement traceability structures*. In Proceedings of the 14th international conference on Software engineering and knowledge engineering, S.135-142, Ischia, Italy, 2002

(Spanoudakis, 2004) Spanoudakis, G., Zisman, A., Perez-Minana, E. und Krause, P.: *Rule-Based Generation of Requirements Traceability Relations*. In Journal of Systems and Software, 72(2): S.105-127, 2004

(Standish, 2004) The Standish Group.: *The CHAOS Report*, 2004.

(Stefanoni, 1994) Stefanoni, M.: *Static analysis for VHDL model evaluation*. In Proceedings of the conference on European design automation, S.586-591, Grenoble, France, 1994

(Strickmann, 2008) Strickmann, J.: *Analysemethoden zur Bewertung von Entwicklungsprojekten - Ein integriertes semantisches Modell von Projekt- und Produktdaten zur Bewertung der Entwicklungsleistung im Projektcontrolling.* Gito Verlag, 2008

(SysML, 2009) OMG Specification: *OMG System Modeling Language (OMG SysML) Specification. Version 1.1*. Final Adopted Specification formal/2008-11-02.

(Tian, 2004) Tian, J.: *Quality-evaluation models and measurements*. In IEEE Software, 21(3): S.84–91, 2004.

(Torroja, 1999) Torroja, Y., Lopez, C., Garcia, M., Riesgo, T., de la Torre, E., Uceda, J.: *ARDID: A Tool for the Quality Analysis of VHDL based Designs*. Second International Forum on Design Languages, 1999

(Toutenburg, 2009) Toutenburg, H. und Knöfel, P.: *Six Sigma – Methoden und Statistik für die Praxis*. Springer, 2.Auflage, 2009

(UML, 2005) OMG Specification. *OMG Unified Modeling Language (OMG UML) Specification. Version 2.0*. Final Adopted Specification formal/2005-07-05

(VDI, 2004) Verein Deutscher Ingenieure - VDI, *VDI 2206 Entwicklungsmethodik für mechatronische Systeme*, 2004

(VMXT, 2009) Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung: *V-Modell XT, Version 1.3*, 2009, Zugriff: http://www.v-modell-xt.de

(von Knethen, 2001a) von Knethen, A.: *A Trace Model for System Requirements Changes on Embedded Systems*. International Workshop on Principles of Software Evolution (IWPSE' 01), 2001

(von Knethen, 2001b) von Knethen, A.: *Change-Oriented Requirements Traceability. Support for Evolution of Embedded Systems*, International Conference on Software Maintenance, Montreal, Quebec, Canada, 2002

(von Knethen, 2002) von Knethen, A.: *Change-oriented Traceability*. *Support for Evolution of Embedded Systems*. Dissertationsarbeit in Experimental Software Engineering, Vol. 9, Fraunhofer IRB Verlag, 2002

(Vörg, 2004) Vörg, A., Radetzki, M. und Rosenstiel, W.: *Measurement of IP qualification costs and benefits*. In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, S. 996- 1001, 2004

(Wagner, 2007a) Wagner, S. und Deißenböck, F. *An Integrated Approach to Quality Modelling*, In Proceedings of the 5th Workshop on Software Quality (WoSQ '07), 2007

(Wagner, 2007b) Wagner, S., Deißenböck, F. und Winter, S.: *Erfassung, Strukturierung und Überprüfung von Qualitätsanforderungen durch aktivitätenbasierte Qualitätsmodelle*. In: Erhebung, Spezifikation und Analyse nichtfunktionaler Anforderungen in der Systementwicklung. Halbtägiger Workshop in Zusammenhang mit der SE Konferenz, 2008

(Wagner, 2008) Wagner, S., Deißenböck, F. und Winter, S.: *Managing Quality Requirements Using Activity-Based Quality Models.* In Proceedings of the 6th Workshop on Software Quality (6-WoSQ), S.29-34, ACM Press, 2008

(Wagner, 2010) Wagner, S., Broy, M., Deißenböck, F., Kläs, M., Liggesmeyer, P., Münch, J. und Streit, J.: *Softwarequalitätsmodelle: Praxisempfehlungen und Forschungsagenda*, Informatik-Spektrum, 33(1), S. 37-44, 2010

(Walenta, 2001) Walenta, T.: *Messbarer Projekterfolg mit der Earned Value-Analyse*. Projekt Magazin, 04/01, 2001

(Wallace, 2001) Wallace, D. und Reeker, L.: *Software Quality*. In *Guide to the Software Engineering Body of Knowledge SWEBOK*, Abram, A. und Bourque, P. (Hrsg.). S.165-184, 2001

(Watkins 1994) Watkins, R. und Nea, M.: *Why and How of Requirements Tracing*, IEEE Software. Vol.11: S. 104-106, 1994

(Wedelstädt, 2001) Wedelstädt, J.: *Grundlagen des Termin- und Kostencontrollings*. Projektmagazin, 2001

(Wildemann, 2003) Wildemann, H.: *Controlling im Total Quality Management*. In: *Qualitätsmanagement im Unternehmen: Grundlagen, Methoden und Werkzeuge, Praxisbeispiele* / Hansen, W.; Kamiske, G. F. (Hrsg.), S. 1-29, Symposion, 2003

(Winter, 2007) Winter, S., Wagner, S. und Deißenböck, F.: *A Comprehensive Model of Usability.* In Proceedings of Engineering Interactive Systems (EIS '07), S.106-122, 2007

(Weinberg, 1991) Weinberg, G.: *Quality Software Management: Systems Thinking*. Dorset House, 1991

(WMC, 2008) Workflow Management Coalition. *Process Definition Interface - XML Process Definition Language*. 2008. Zugriff: http://www.wfmc.org/index.php?option=com_docman&task=doc_details&gid=40&Itemid=126

(Zadeh, 1965) Zadeh, L.H.: *Fuzzy Sets*. Information and Control, vol.8, S. 338-353, 1965

(Zisman, 2002) Zisman, A., Spanoudakis, G., Perez-Minana, E. und Krause P.: *Towards a Traceability Approach for Product Families Requirements*, Proceedings of 3rd ICSE Workshop on Software Product Lines: Economics, Architectures, and Implications, Orlando, USA, 2002

(Zisman, 2003) Zisman, A., Spanoudakis, G., Perez-Minana, E. und Krause P.: *Tracing Software Requirements Artefacts*. The 2003 International Conference on Software Engineering Research and Practice (SERP'03), Las Vegas, USA, 2003

(Zollondz, 2006) Zollondz, H.D.: Grundlagen des Qualitätsmanagements, 2.Auflage 2006

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe.

Stefan Häusler