



# **Softwaregrößenmessung im Kontext von Software- Prozessbewertungsmodellen**

***Diplomarbeit***

*von Karsten Richter*

# Index

<b>INDEX .....</b>	<b>2</b>
<b>ABBILDUNGSVERZEICHNIS.....</b>	<b>4</b>
<b>TABELLENVERZEICHNIS .....</b>	<b>5</b>
<b>1 EINLEITUNG.....</b>	<b>7</b>
1.1 Motivation .....	7
1.2 Zielstellung .....	9
1.3 Aufbau der Arbeit.....	9
<b>2 GRUNDLAGEN.....</b>	<b>11</b>
2.1 Software Engineering und Softwareprozessbewertung.....	11
2.2 Software-Produkt-Qualität .....	15
2.3 Prozess-/Ressourcen-Qualität .....	16
2.4 Life-Cycle-Modelle .....	19
2.4.1 Sequenzielle Modelle.....	19
2.4.2 Nicht-sequenzielle Modelle .....	22
2.5 Zusammenfassung .....	22
<b>3 BETRACHTUNGEN EINZELNER PROZESSBEWERTUNGSMODELLE .....</b>	<b>24</b>
3.1 ISO 9001:2000 .....	24
3.2 CMM.....	26
3.3 CMMI .....	29
3.3.1 CMMI – staged .....	30
3.3.2 CMMI – continuous .....	33
3.3.3 Überführung.....	35
3.4 Unterschiede / Gemeinsamkeiten der einzelnen Modelle.....	36
3.5 Zusammenfassung .....	38
<b>4 EINFÜHRUNG IN SIZE-MEASUREMENT .....</b>	<b>40</b>
4.1 Allgemeine Betrachtungen .....	40
4.2 Physikalische Größe .....	41
4.2.1 Lines of Code.....	41
4.2.2 Halstead .....	42

<b>4.3</b>	<b>Funktionale Größe von Software .....</b>	<b>43</b>
4.3.1	Albrechts Ansatz .....	45
4.3.2	Full Function Point .....	47
4.3.3	Weitere Ansätze .....	48
<b>4.4</b>	<b>Zusammenfassung .....</b>	<b>49</b>
<b>5</b>	<b>FSM &amp; SW-LEBENSZYKLUS .....</b>	<b>50</b>
<b>5.1</b>	<b>Wann weiß man viel über die Softwaregröße .....</b>	<b>50</b>
<b>5.2</b>	<b>Messung vs. Schätzung .....</b>	<b>52</b>
<b>5.3</b>	<b>Einfluss von Life-Cycle-Modellen .....</b>	<b>53</b>
5.3.1	Aktuelle funktionale Größe .....	54
5.3.2	Eigenschaften des Endproduktes .....	55
<b>5.4</b>	<b>Zusammenfassung .....</b>	<b>57</b>
<b>6</b>	<b>FSM &amp; CMMI .....</b>	<b>59</b>
<b>6.1</b>	<b>Metriken pro CMMI-Stufe .....</b>	<b>59</b>
<b>6.2</b>	<b>FSM-basierte Metriken .....</b>	<b>59</b>
<b>6.3</b>	<b>FSM-basierte Unterstützung für das CMMI .....</b>	<b>60</b>
6.3.1	Level 1 .....	61
6.3.2	Level 2 .....	61
6.3.3	Level 3 .....	70
6.3.4	Level 4 .....	79
6.3.5	Level 5 .....	82
6.3.6	Zusammenfassung .....	85
<b>7</b>	<b>CMMI &amp; LIFE-CYCLE .....</b>	<b>89</b>
<b>7.1</b>	<b>Zusammenfassung .....</b>	<b>93</b>
<b>8</b>	<b>PRAKTISCHER ANSATZ .....</b>	<b>94</b>
<b>8.1</b>	<b>Vorbetrachtungen und Unternehmensbewertung .....</b>	<b>94</b>
<b>8.2</b>	<b>Erstellen einer FSM-basierten Messumgebung für CMMI-Level 2 .....</b>	<b>95</b>
<b>ANHANG A</b>	<b>.....</b>	<b>99</b>
<b>ANHANG B</b>	<b>.....</b>	<b>108</b>
<b>9</b>	<b>REFERENZEN .....</b>	<b>124</b>

## Abbildungsverzeichnis

Abbildung 1:	Prozessbewertungsmodelle .....	8
Abbildung 2:	Softwaremanagement .....	12
Abbildung 3:	Software-Qualitätsmanagement .....	14
Abbildung 4:	Qualität nach ISO 9126 .....	15
Abbildung 5:	Qualität nach DIN 66272 .....	16
Abbildung 6:	Zusammenhang zwischen Prozess und Ressource.....	17
Abbildung 7:	Wasserfallmodell.....	20
Abbildung 8:	V-Modell .....	21
Abbildung 9:	Spiralmodell .....	22
Abbildung 10:	CMM-Stufen .....	26
Abbildung 11:	CMMI-Stufen .....	30
Abbildung 12:	Struktur der CMMI-Stufenform .....	32
Abbildung 13:	Entwicklung der funktionalen Größenmessung .....	44
Abbildung 14:	Parameter der funktionalen Größenmessung .....	45
Abbildung 15:	Layer-Prinzip von COSMIC FFP.....	48
Abbildung 16:	Modell für Konsistenzerhalt bei funktionaler Größenmessung .....	51
Abbildung 17:	Wissensentwicklung mit Projektfortschritt .....	51
Abbildung 18:	Modell zur Kalibrierung von Schätzungen und Messungen .....	53
Abbildung 19:	Schätzen/Messen der aktuellen funktionalen Größe .....	55
Abbildung 20:	Entwicklung von Schätzwerten im Lebenszyklus.....	56
Abbildung 21:	Schätzen/Messen der finalen funktionalen Größe.....	56
Abbildung 22:	Kalibrierung der Messungen im Lebenszyklus .....	57
Abbildung 26:	FSM-Unterstützung für CMMI-Level 5.....	85
Abbildung 27:	FSM-Unterstützung pro KPA.....	86
Abbildung 28:	Unterstützung der CMMI-Level durch FSM .....	87
Abbildung 29:	Zuwachs der FSM-Unterstützung .....	88
Abbildung 30:	Zusammenhang Lifecycle - CMMI.....	89
Abbildung 31:	Mapping von KPAs auf Lifecycle-Elemente .....	91
Abbildung 32:	Zusammenhang CMMI – Lifecycle .....	92
Abbildung 33:	FSM-Unterstützung für CMMI-Level 2.....	95

## Tabellenverzeichnis

Tabelle 1:	CMMI-Level mit zugehörigen Schlüsselprozessarealen.....	31
Tabelle 2:	Zuordnung der Schlüsselprozessareale zu CMMI-Kategorien .....	34
Tabelle 3:	Benötigter Fähigkeitsgrad für Level 2 .....	35
Tabelle 4:	Benötigter Fähigkeitsgrad für Level 3 .....	36
Tabelle 5:	Benötigter Fähigkeitsgrad für Level 4 .....	36
Tabelle 6:	Benötigter Fähigkeitsgrad für Level 5 .....	36
Tabelle 7:	Vergleich von CMMI und DIN EN ISO 9001:2000 .....	38
Tabelle 8:	LOC-Varianten.....	42
Tabelle 9:	Wichtungstabelle für UFC-Bestimmung.....	46
Tabelle 10:	Kostentreiber für technischen Komplexitätsfaktor .....	46
Tabelle 11:	Wichtungstabelle für techn. Komplexitätsfaktor .....	46
Tabelle 12:	Kategorien bei FSM-Unterstützungsuntersuchungen .....	61
Tabelle 13:	projektunterstützende vs. unternehmensunterstützende KPAs .....	90
Tabelle 14:	Metriken für CMMI-Level 2 .....	97

## ***Diplomaufgabenstellung Karsten Richter***

**Arbeitsthema:** Größenmessung von Software im Kontext der Softwareprozessbewertung

Die Bearbeitung der Aufgabenstellung ist in folgenden Teilschritten zu realisieren:

- Betrachtung/Diskussion von Prozessbewertungsmodellen und Auswahl eines geeigneten Modells für die Bewertung von Softwareentwicklungsprojekten
- Betrachtung von Ansätzen zur Größenmessung von Software und Auswahl eines geeigneten Größenmaßes
- Mapping zwischen dem gewählten Größenmaß und dem favorisierten Prozesswertungsmodell und Diskussion der Eignung des Maßes für die Unterstützung der Prozessbewertung
- Diskussion der Zusammenhänge zwischen dem gewählten Größenmaß, dem Prozessbewertungsmodell so wie dem Softwarelebenszyklus
- Nach Möglichkeit: Herleitung eines praktischen Ansatzes/Vorschlages für die Umsetzung eines Größenbasierten Prozessbewertungsansatzes bei der ICUBIC AG

# 1 Einleitung

## 1.1 *Motivation*

Die heutige Zeit ist von einem enormen Anstieg des Wettbewerbs in allen Industriebereichen gekennzeichnet. Dieser zwingt Unternehmen dazu, qualitativ hochwertige Produkte anbieten zu können. Eine Firma die nicht fähig ist, diesen Anforderungen gerecht zu werden, kann sich gegen eventuell vorhandene Konkurrenten nicht behaupten. Deshalb sollte die Verbesserung der Produktqualität in jedem Unternehmen die höchste Priorität besitzen.

Ein Weg zur Verbesserung der Produktqualität ist die Steigerung der Güte der zur Herstellung verwendeten Ressourcen und Prozesse. Zum Zweck dieser Verbesserung wurde bereits in der Vergangenheit eine Vielzahl von Prozessbewertungsmodellen erarbeitet, die aber ebenso verwendet werden können, um die Qualitätssteigerungsvorgänge koordinieren und bewerten zu können.

Durch die immer weiter fortschreitende Weiter- und Neuentwicklung derartiger Modelle wurden im Lauf der Zeit sehr viele verschiedene Modelle erarbeitet. Diese zum Teil aufeinander aufbauenden oder sich gegenseitig unterstützenden Versionen befassen sich dabei mit den unterschiedlichsten Ausrichtungen und Schwerpunkten der zu bewertenden und zu verbessernden Entwicklungsprozesse. Die folgende Abbildung nach [Quagmire2005] soll dem Leser in diesem Zusammenhang einen kleinen Überblick über die bereits entwickelten Modelle und deren Zusammenhänge näher bringen.

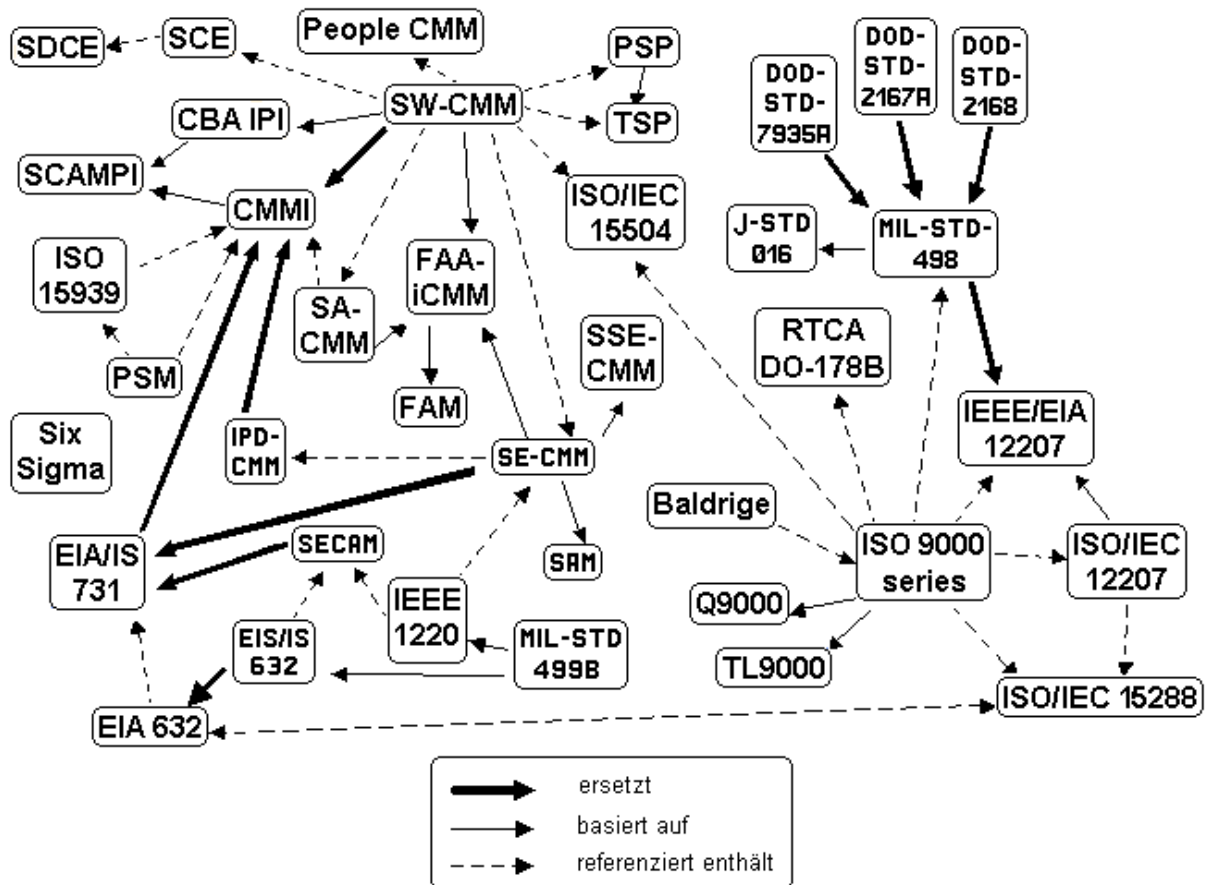


Abbildung 1: Prozessbewertungsmodelle

Hier ist bereits zu erkennen, dass sich über die Zeit eine nahezu unüberschaubare Menge derartiger Modelle entwickelt hat.

Einige der hier dargestellten Ansätze sind auf diverse Industriezweige anwendbar, andere hingegen weisen lediglich spezielle Ausrichtungen auf. Diese Arbeit ist aufgrund des konkreten Umfeldes auf Softwareentwicklung ausgerichtet, und betrachtet deshalb nur softwarerelevante Aspekte der jeweiligen Modelle.

Im Zusammenhang mit der Prozessbewertung wurden auch weitere Kerngedanken bei der systematischen Produktentwicklung weiterentwickelt. Einen Gedanken der hierbei von besonderem Interesse ist, stellt die Softwaremessung dar. Die zu diesem Zweck eingeführten Metriken und Messansätze haben sich im Lauf der Zeit nicht minder stark entwickelt. Dabei überschreitet die Anzahl der Herangehensweisen zur Bestimmung der Softwaregröße die Menge der Prozessbewertungsmodelle bei Weitem. Die ist beispielsweise an einem durch [Zuse2003] entwickelten Informationssystem erkennbar. Es enthält neben weiteren Informationen eine Kollektion von ca. 1500 Metriken. Diese stellen allerdings ebenso nur einen Ausschnitt der Möglichen Metriken dar.

## **1.2 Zielstellung**

Ein Anliegen dieser Ausarbeitung soll es sein, einen Überblick über drei der wohl meistgenutzten Prozessbewertungsmodelle zu geben, und deren Eignung zur Anwendung in Softwareunternehmen zu prüfen. Die in diesem Zusammenhang näher untersuchten Ansätze sind dabei die Modelle CMM, CMMI und DIN EN ISO 9001:2000.

Ein weiterer Kerngedanke ist die Betrachtung einiger der gebräuchlichsten Methoden zur Bestimmung der Softwaregröße. Ansätze die in diesem Zusammenhang untersucht werden sind Lines of Code, Halstead und die funktionale Größenmessung. Die Eignung dieser Metriken für die Unterstützung der Prozessbewertungsmodelle soll dabei überprüft und auf der Basis dieser Überlegungen ein geeignetes Maß für die weitergehenden Betrachtungen gewählt werden.

Das Hauptanliegen dieser Arbeit soll es allerdings sein, eine Verbindung zwischen den beiden erwähnten Ansätzen zur Prozessverbesserung und –Bewertung aufzuzeigen. Von besonderem Interesse ist dabei, in welcher Art und Weise ein durchgehendes Größenmaß die Bewertung beziehungsweise Verbesserung von Software-Prozessen unterstützen kann.

## **1.3 Aufbau der Arbeit**

Zur Erfüllung des Anliegens dieser Arbeit, werden im Vorfeld einige allgemeine Betrachtungen zum Software Engineering und zur Softwareprozessbewertung durchgeführt werden. Diese münden in die Vorstellung von drei der weltweit wohl meistgenutzten Prozessbewertungsmodelle. Im Einzelnen sind dies das Capability Maturity Model CMM, dessen Weiterentwicklung, das Capability Maturity Model Integration, und der internationale Industriestandard DIN EN ISO 9001:2000. Das Aufzeigen der Vor- beziehungsweise Nachteile dieser Ansätze ist dabei ebenso ein Hauptanliegen der Untersuchungen, wie die Bewertung dieser Ansätze hinsichtlich ihrer Eignung zur Einschätzung von Softwareentwicklungsprozessen. Ein ebenfalls in diesem Abschnitt aufgezeigter Ansatz zur strukturierten Entwicklung von Software sind die Lebenszyklusmodelle. Der Zusammenhang zwischen der Softwareprozessbewertung und diesen Modellen soll allerdings in späteren Betrachtungen näher beleuchtet werden.

Im Anschluss an diese Ausführungen, werden Betrachtungen einiger Ansätze zur Bestimmung der Größe von Software in den Fokus der Arbeit rücken. Hierbei finden nach

einigen grundlegenden Überlegungen, vor allem Lines of Code, die Überlegungen von Halstead und die funktionale Größenmessung verstärkte Beachtung. Die Eignung der dargestellten Metriken hinsichtlich einer durchgehenden Unterstützung während des Entwicklungsprozesses stellt dabei den abschließenden Gedanken dieses Abschnittes dar.

Mit der Zuordnung von Metriken zu dem bereits im Vorfeld gewählten Prozessbewertungsmodell befassen sich die darauf folgenden Abschnitte. Die Unterstützung der dort dargestellten Messansätze durch das gewählte Maß stellt dabei einer der verfolgten Hauptgedanken.

Resultierend aus diesen Überlegungen werden Betrachtungen zu den Zusammenhängen zwischen dem gewählten Größenmaß, dem Prozessbewertungsmodell und den bereits im Vorfeld erläuterten Lebenszyklusmodellen den Abschluss der theoretischen Überlegungen bilden.

Die praktische Anwendbarkeit der in den vorhergehenden Betrachtungen eingeführten Zusammenhänge, soll den Kern des letzten Abschnittes bilden. Dazu wird nach Möglichkeit ein Ansatz zur Prozessverbesserung am Beispiel der icubic AG eingeführt. Dieser soll aufzeigen, auf der Basis welcher Messansätze die Verbesserung im Rahmen des gewählten Prozessbewertungsmodells ermöglicht wird. Damit soll die theoretische Grundlage für die praktische Einführung der entsprechenden Vorgehensweisen geschaffen werden.

## **2 Grundlagen**

Software wird in der heutigen Zeit zu immer komplexeren Strukturen verwoben. Diese Konstrukte lassen sich intuitiv nur schwer erstellen oder managen. Eine in diesem Zusammenhang häufig angewandte Methodik basiert auf der Übernahme von Tätigkeiten aus der Industriellen Fertigung von Produkten. Die Rede ist vom Software-Engineering. Neben allgemeinen Betrachtungen zu dieser Thematik befassen sich die folgenden Abschnitte mit tiefergehenden Untersuchungen der Einzelgebiete des Software-Engineering.

### **2.1 Software Engineering und Softwareprozessbewertung**

Laut [Balzert 2000] stellt Software-Engineering „die zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Herstellung und Anwendung von umfangreichen Software-Systemen“ dar. Bereits 1969, als der Begriff „Software-Engineering“ geprägt wurde, war aufgrund der Namensgebung anscheinend die Tendenz bereits absehbar, dass sich die Entwicklung von Software mit ingenieurwissenschaftlichen Methoden qualitätssteigernd auswirken kann.

Mit der Anwendung ingenieurmäßiger Methoden zur Softwareentwicklung geht auch das Software-Management einher. Wie der Name schon sagt, befasst es sich mit dem Management von verschiedenen Bereichen der Softwareentwicklung. Die folgende Abbildung nach [DumkeSE2] gibt einen Überblick über die einzelnen Gebiete, die Gegenstand des Software-Managements sind.

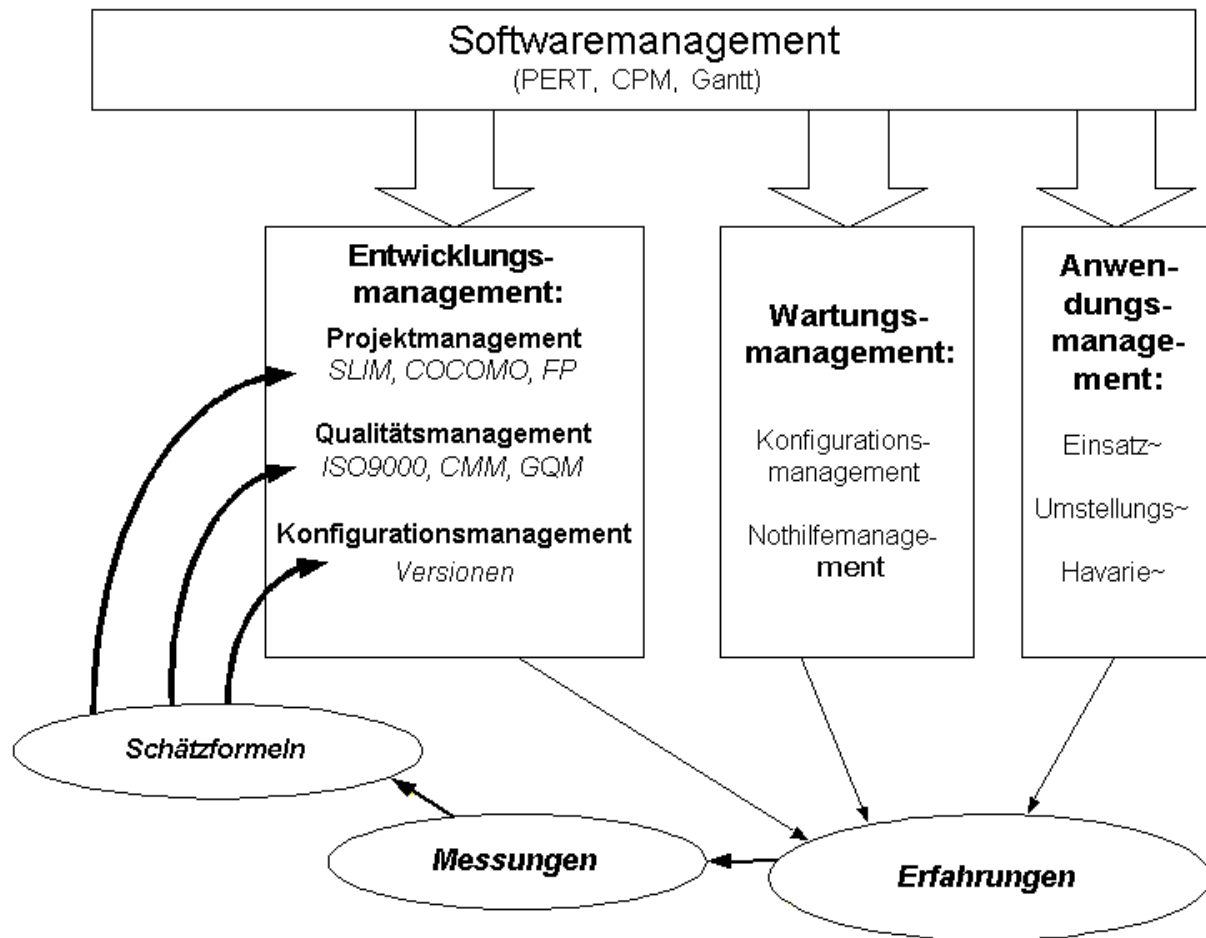


Abbildung 2: Softwaremanagement

Im Rahmen dieser Arbeit ist allerdings lediglich das Entwicklungsmanagement von Interesse. Im speziellen auch nur ein Aspekt dieses Abschnitts, das Software-Qualitätsmanagement, wobei Einflüsse der anderen Managementformen natürlich nicht ausgeschlossen und gegebenenfalls mit berücksichtigt werden.

Im Vorfeld der Betrachtung des Software-Qualitätsmanagements muss man sich allerdings mit dem Begriff der Qualität im Allgemeinen befassen. Das Problem bei der Festlegung, was Qualität eigentlich darstellt, liegt in der Diskrepanz zwischen der alltäglichen Bedeutung und der Nutzung in professioneller Hinsicht. Die in [Kan2003] dargestellte populäre Ansicht, Qualität sei eine schwer greifbare Charakteristik, welche gefühlt, diskutiert und darauf basierend subjektiv bewertet werden kann, ist für den Gebrauch in der Industrie natürlich wenig effizient. Sie weist auf die allgemeine Ansicht hin, dass Qualität nicht kontrolliert oder gemanagt werden kann. Eine weitere in diesem Zusammenhang dargestellte Sichtweise ist die

Herangehensweise, dass der Preis, die Raffinesse oder höhere Komplexität eines Produktes Eigenschaften der Qualität dieses Produktes darstellen.

Diese beiden Vorstellungen von Qualität sind mit der industriellen Sichtweise natürlich nicht vereinbar. Allerdings existieren auch in der professionellen Ansicht über Qualität unterschiedliche Ansichten. Zum einen stellt Qualität nach [Crosby1979] die „Übereinstimmung mit den Anforderungen“ dar, auf der anderen Seite allerdings nach [Juran1970] die „Einsatztauglichkeit“. Diese beiden Ansichten sind allerdings im Gegensatz zur populären Herangehensweise weitestgehend miteinander vereinbar. Das beruht darauf, dass die Einsatztauglichkeit auf der Erfüllung von Kundenbedürfnissen beruht, welche auch in den Anforderungen an das Produkt wiedergefunden werden können. Die Übereinstimmung mit den Anforderungen bestimmt also in einem großen Maße die Einsatztauglichkeit bezüglich einer festgelegten Nutzergruppe. Die Nähe zum Konsumenten wird im Abschnitt über die Prozess-/Ressourcen-Qualität noch stärker in Erscheinung treten.

Eine allgemeiner gefasste Definition des Begriffes Qualität kann auch in verschiedenen Normen wiedergefunden werden. Beispielsweise stellt Qualität laut [DIN8402] die

„Gesamtheit von Merkmalen (und Merkmalswerten) einer Einheit bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen“

dar. Qualität basiert also auf festgesetzten Eigenschaften eines Objekts. Das Qualitätsmanagement befasst sich in diesem Zusammenhang mit der Definition, dem Erreichen bzw. Einhalten eben dieser Eigenschaften. Zu diesem Zweck werden je nach Anwendungsbereich spezielle Techniken und Methoden eingesetzt.

Die vorhergehenden Eigenschaften sind allerdings nicht exklusiv auf industriell gefertigte Produkte zu beziehen. Auch eine Ausweitung auf die Entwicklung von Softwareprodukten lässt sich ableiten. Daher kann das Software-Qualitätsmanagement auch als ein spezielles Gebiet des Qualitätsmanagement verstanden werden. Nach [DumkeSE2] stellt es „die Sicherung von Qualitätsmerkmalen für das Software-Produkt auf der Grundlage der Prozess- und Ressourcenqualität durch organisatorische Methoden und Maßnahmen unter Anwendung spezieller Techniken und Technologien“ dar.



## 2.2 Software-Produkt-Qualität

Die Qualität eines Produktes lässt sich anhand verschiedener Kriterien bestimmen. Dies ist sowohl bei industriell gefertigten Erzeugnissen als auch bei Softwareprodukten der Fall. Allerdings können normale Kriterien weitestgehend nicht auf Software angewendet werden. Auf welchen Parametern Softwarequalität beruht und wie sie aufrechterhalten oder verbessert werden kann, ist der Kerngedanke dieses Abschnittes.

Die Software-Produkt-Qualität stellt die Qualität eines Softwareproduktes dar. Basierend auf einer Vielzahl von Gesichtspunkten wird sie unter anderem durch Software-Qualitätsmodelle beschrieben. Ein solches Modell ist beispielsweise die ISO 9126, welche die in folgender Abbildung nach [ZusePaper] dargestellten Aspekte von Software-Produkt-Qualität beinhaltet.

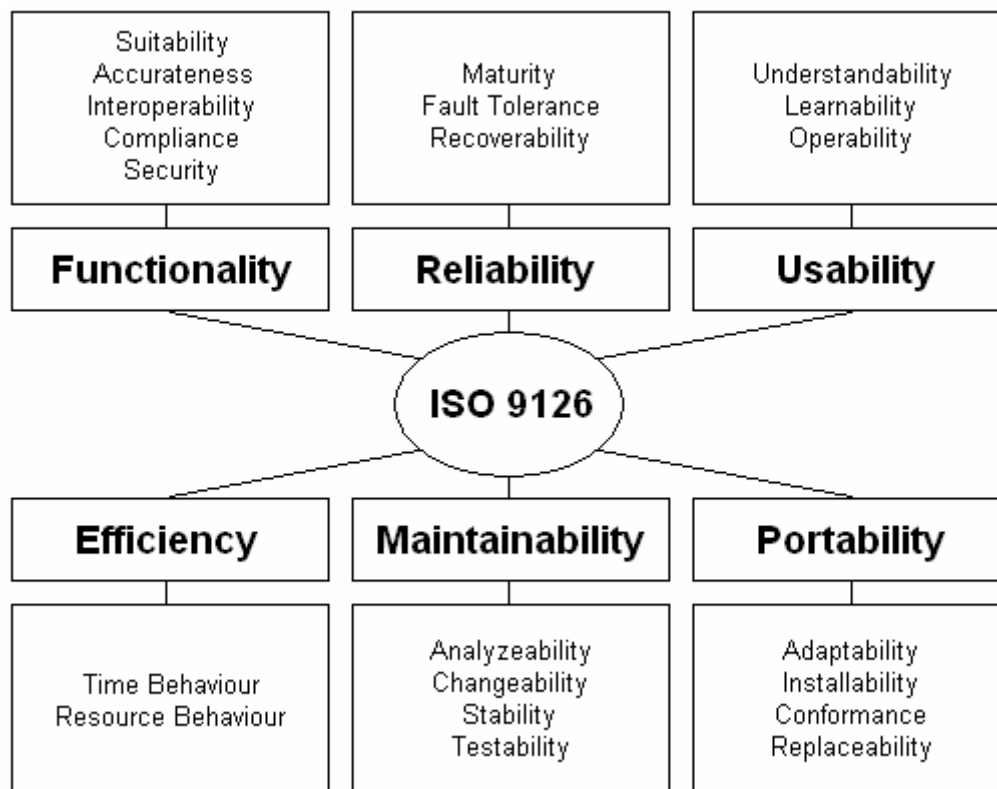


Abbildung 4: Qualität nach ISO 9126

Diese Herangehensweise wird ebenso innerhalb des deutschsprachigen Ansatzes in Form der DIN 66272 aufgegriffen. Die Ähnlichkeit beider Ansätze lässt sich durch das Betrachten der obigen und der folgenden Abbildung [DumkeSE2] leicht erkennen:

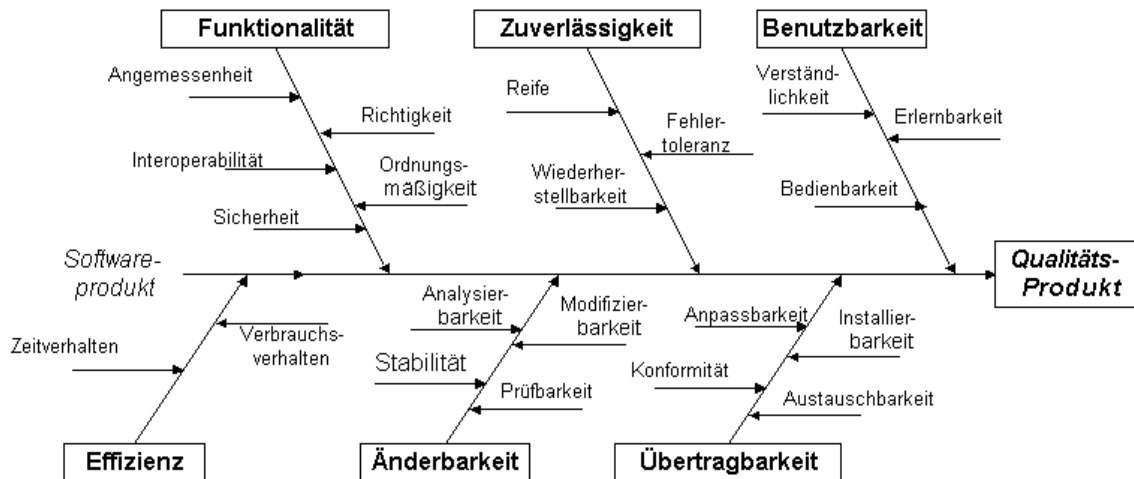


Abbildung 5: Qualität nach DIN 66272

Weiterhin existiert noch eine Vielzahl weiterer Modelle, die für spezielle Umgebungen entwickelt worden sind. Hierbei sei für weitergehende Untersuchungen auf SPARDAT ([Wallmüller2001]), McCall ([McCall1977]) und Boehm ([Boehm1976]) verwiesen.

Diese Software-Qualitätsmodelle zeigen also verschiedene qualitative Eigenschaften von Software auf. Für die effiziente Planung und Lenkung der Produktentwicklung bei gleichzeitiger Einhaltung eben dieser qualitativen Anforderungen an ein Softwareprodukt hat sich laut [WU-Wien] eine Zerlegung des Gesamtprojektes in mehrere kleinere Teilprojekte als effektiv erwiesen. Diese Teilprozesse werden nacheinander durchlaufen und erlauben somit eine effektivere Kontrolle des Geschehens im Verlauf der Softwareentwicklung. Die Aufgliederung in einzelne Teilprozesse ermöglicht ebenso eine variabelere Anordnung dieser Abschnitte. Daraus ergeben sich mehrere verschiedene Zusammenstellungen, welche auch Lebenszyklus-Modelle genannt werden. Auf die Betrachtung einiger dieser Modelle soll zu einem späteren Zeitpunkt noch ein verstärktes Augenmerk gelegt werden.

Das nun folgende Kapitel soll sich zuvor mit dem bereits in Abbildung 3 gezeigten zweiten Standpfeiler des Software-Qualitätsmanagements, der Prozess- und Ressourcenqualität, auseinandersetzen.

### 2.3 Prozess-/Ressourcen-Qualität

Wie bereits festgehalten stellen sowohl die Prozess- als auch die Ressourcen-Qualität indirekte Einflussfaktoren der Produktqualität dar. Sie sind allerdings trotzdem nicht zu unterschätzende Qualitätsfaktoren. Häufig wird die Prozessqualität auch in Zusammenhang

mit der Ressourcenqualität gesetzt. Diese Relation zwischen beiden Qualitätsformen wird als Qualitätsfähigkeit bezeichnet. In dieser Arbeit werden sie ebenso nicht losgelöst voneinander betrachtet, da Standards für die Prozessbewertung im Vordergrund stehen sollen, welche im Allgemeinen das Ressourcen-Management mit einschließen. Dazu aber später mehr.

Die Basis dieses Abschnitts sollen grundlegende Betrachtungen zur Prozess- bzw. Ressourcen-Qualität bilden. Dabei ist es unabdingbar sich über die Bedeutung der Begriffe Prozess und Ressource klar zu sein. Bei diesen Überlegungen wird auch bereits deutlich warum beide Begriffe häufig nicht getrennt voneinander betrachtet werden. Nach [Fenton1997] stellt ein Prozess im Bereich der Softwareentwicklung eine „Sammlung von Software-verwandten Aktivitäten“ dar. Ressourcen werden dort als „von einer Prozessaktivität benötigte Einheiten“ definiert, was bedeutet, dass es sich dabei um im Verlauf von Prozessen einzusetzende personelle und materielle Mittel handelt.

In [Fenton1997] wird in diesen Zusammenhang zwischen Prozess und Ressource auch das Produkt mit eingebunden. Beschrieben wird das Produkt hier als „alle Artefakte, Ergebnisse und Dokumente, welche aus einer Prozessaktivität resultieren“. Die Verbindung zwischen diesen drei Punkten wird noch einmal anhand folgender Abbildung verdeutlicht.

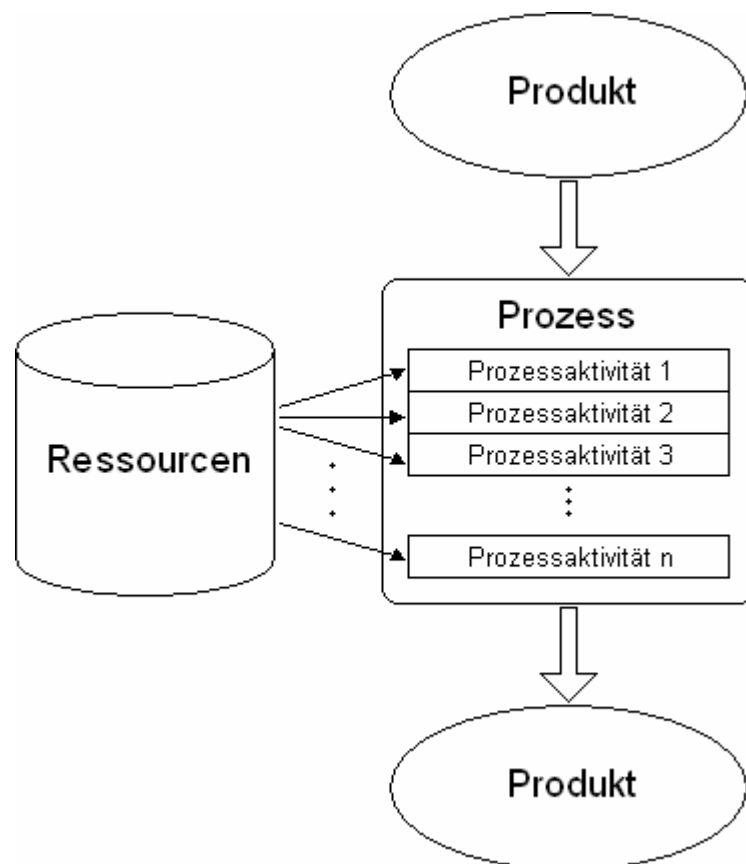


Abbildung 6: Zusammenhang zwischen Prozess und Ressource

Das Hauptziel der Bestimmung der Prozessqualität ist nach [DumkeSE2] die „Sicherung der Produktqualität“. Weitere Ziele, die sich hiermit verbinden lassen, sind beispielsweise Kostensenkung oder eine Verringerung der Produktionszeit.

Zur Bestimmung der Prozessqualität können, ebenfalls nach [DumkeSE2], zwei unterschiedliche Typen von Anzeichen dienen:

- direkte Indikatoren
  - o Kosten
  - o Entwicklungszeit
  - o Benötigte personelle / materielle Ressourcen
  - o ...
- indirekte Indikatoren
  - o Entwicklungskomplexität
  - o Time-to-Market
  - o Risiko
  - o ...

Die einzelne Überprüfung aller möglichen Eigenschaften von Prozessen mit dem Ziel ihrer Verbesserung ist sehr zeitaufwendig. Speziell für diesen Zweck entwickelte Prozessbewertungs-Standards bieten hier eine hervorragende Ausgangsbasis für die Bestimmung der Prozessqualität und deren Verbesserung. Sie stellen Zusammenfassungen der für eine hohe Produktqualität notwendigen Prozesse dar. Auf diese Normen soll allerdings in einem späteren Abschnitt noch näher eingegangen werden.

Die Qualität der im Entwicklungsprozess verwendeten Ressourcen spielt eine in die Prozessqualität eingebettete Rolle. Ein verdeutlichendes Beispiel dafür ist, dass ein Prozess welcher von ungenügend geschultem Personal oder mit minderwertigen Materialien durchgeführt wird, kaum in der Lage sein wird, ein qualitativ hochwertiges Produkt herzustellen. Ein zur Wahrung der Produktqualität unabdingbarer Gegenstand ist also auch die Sicherung der Ressourcenqualität.

Nach [Fenton1997] werden Ressourcen gemessen, um in der Lage zu sein, deren benötigten Bedarf, ihre Kosten, Qualität oder Produktivität abschätzen zu können. Im Zusammenhang mit dem letzten Punkt, wird die Messung weiterer Faktoren angeraten. Beispielsweise sind dies Alter, Erfahrung oder Intelligenz für personelle Ressourcen praktikable Ansätze für weitere Bestimmungen.

## **2.4 Life-Cycle-Modelle**

Wie bereits im Abschnitt „2.2 Software-Produkt-Qualität festgehalten wurde, ist eine Zerlegung eines Entwicklungsprozesses in mehrere kleinere Teilprozesse sinnvoll. Mithilfe dieser Abschnitte lassen sich dann Interaktionen und Datenflüsse innerhalb des Gesamtprozesses darstellen. Aufgrund verschiedener Ansätze hat sich in diesem Feld ebenfalls eine Vielzahl verschiedener Modelle entwickelt.

Die Basiselemente all dieser Modelle werden in vielen Quellen als die folgenden zusammengefasst:

- Anforderungsanalyse
- Spezifikation
- Design
- Implementierung
- Testung
- Auslieferung

Diese Elemente werden zu den verschiedenen Modellen kombiniert und dabei in unterschiedlicher Art und Weise feiner unterteilt oder zusammengefasst. Ebenso verschieden werden hierbei die Relationen zwischen diesen Phasen aufgeschlüsselt. In diesem Kapitel soll nun ein Überblick über einige der meistverwendeten Modelle gegeben und deren speziellen Eigenschaften untersucht werden. Eine allgemeine Unterscheidung kann hierbei allerdings bereits aufgrund ihrer Vorgehensweise getroffen werden. So wird im Allgemeinen zwischen sequenziellen und nicht-sequenziellen Modellaufbauten unterschieden.

### **2.4.1 Sequenzielle Modelle**

Sequenzielle Anordnungen zeichnen sich durch die Eigenschaft aus, dass sie innerhalb eines Zyklus weitestgehend lediglich einmal durchlaufen werden. Es existieren jedoch meist Rücksprungmechanismen die ein Zurückkehren in frühere Phasen und damit eine Anpassung der bereits festgesetzten Entwicklungsparameter ermöglichen.

Die ursprüngliche Anordnung der bereits herausgearbeiteten Einzelabschnitte ist annähernd im Wasserfall-Modell zu finden. Die folgende Darstellung hierzu entstammt [Rook1991] und zeigt das Modell entsprechend einer Darstellung nach Royce aus dem Jahr 1970:

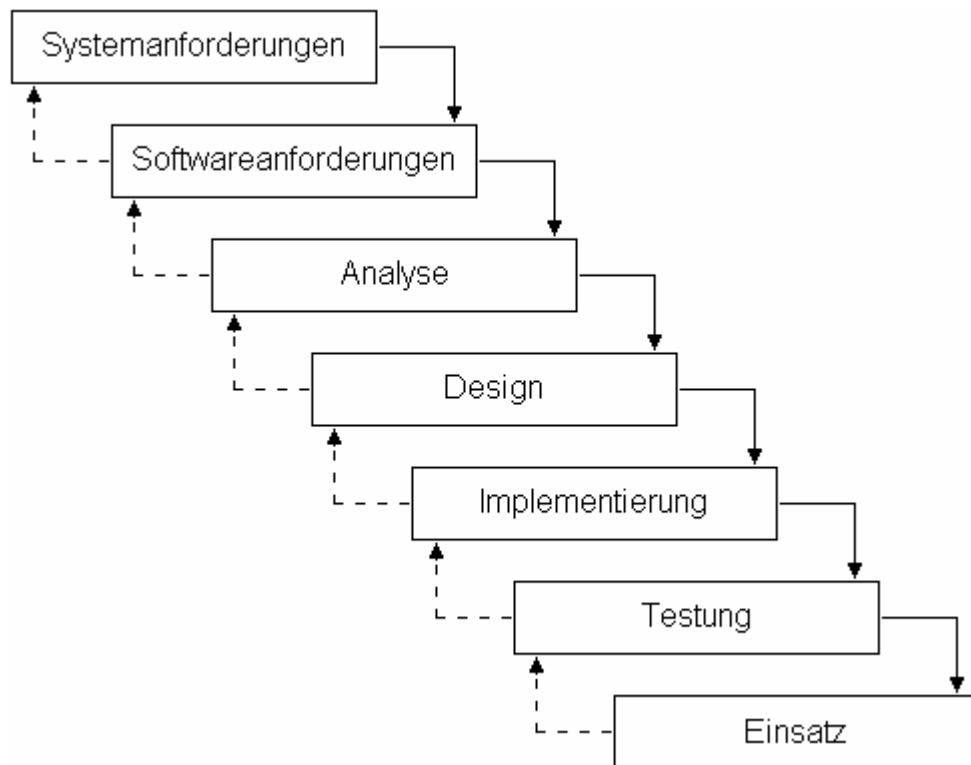


Abbildung 7: Wasserfallmodell

In dieser Anordnung der Teilprozesse ist erkennbar, dass jede Phase theoretisch nur einmal im Verlauf der Produktentwicklung durchlaufen wird und die Ergebnisse oder auch Produkte des vorhergehenden Abschnitts die Eingabedaten des folgenden Abschnitts sind. Die gestrichelten Pfeile stellen dabei kontrollierte Iterationen zwischen den Phasen dar. Das bedeutet, dass neben der gerichteten Abfolge der Abschnitte auch Rücksprünge zwischen den Ebenen möglich sind. Dies kann beispielsweise der Fall sein, wenn während der Designphase ein Fehler in den Analysedokumenten festgestellt wird oder Fehler der Implementierung in der Testphase entdeckt werden.

Ein weiteres sequenzielles Modell ist das V-Modell. Es wird als Entwicklungsstandard für IT-Systeme bei den Bundesbehörden eingesetzt. Die folgende Darstellung nach [Rook1991] zeigt eine mögliche Form eines solchen Modells. Diese Version ist allerdings nicht die Originalversion, sondern enthält bereits die Produkte, die aus den jeweiligen Teilabschnitten resultieren und zeigt gleichzeitig die Verknüpfungen zwischen den einzelnen Tests und den Dokumenten, welche, aus den frühen Phasen resultierend, die Basis dieser Überprüfungen bilden.

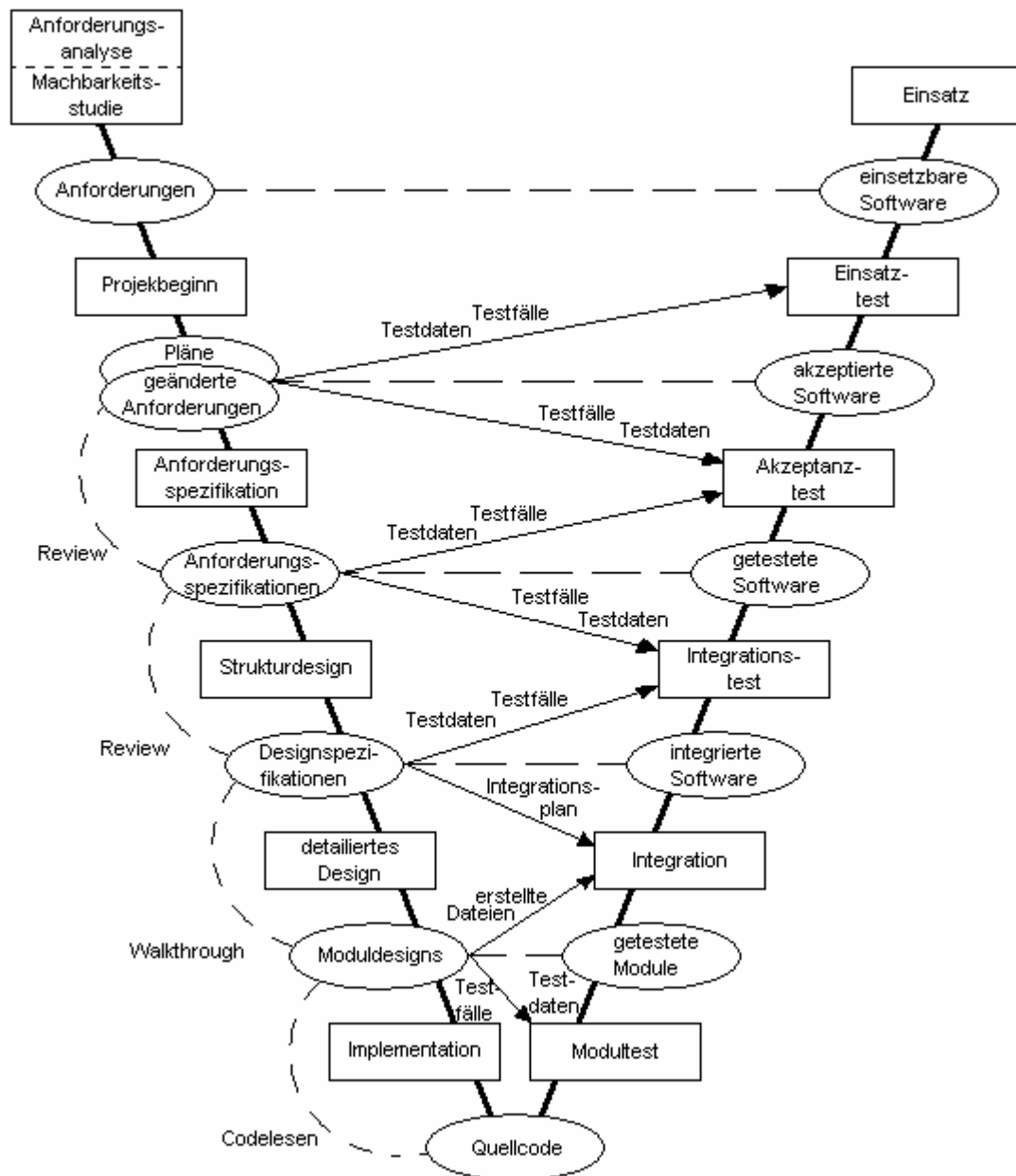


Abbildung 8: V-Modell

An dieser Modelldarstellung ist sehr gut erkennbar, in welcher Art und Weise die innerhalb eines Lebenszyklusmodells genauer spezifiziert werden können. So ist hier neben den bereits genannten Produkten der jeweiligen Teilabschnitte bereits eine Vielzahl der zwischen den einzelnen Elementen vorherrschenden Beziehungen dargestellt. Im Detail sind dies die auf der linken Seite dargestellten Überprüfungsmethoden vorhergehender Abschnitte (Review, Walkthrough, Codelesen) und die Informationen darüber, wo Daten aus den links angeordneten Phasen in den späteren, rechts aufgezeigten, wiederzufinden sind.



Prozessqualität von gesteigertem Interesse. Die Betrachtungen zu diesen Relationen münden in Untersuchungen, in welcher Art und Weise sich die verschiedenen Qualitätsausprägungen untereinander beeinflussen und unterstützen können. In diesem Zusammenhang wurde aufgezeigt, dass jede in der Lage ist, die Qualitätsfähigkeit eines Unternehmens im Bezug auf die Wahrung oder Erhöhung der Produktqualität zu unterstützen.

Weiterführend wurden anschließend Life-Cycle-Modelle zur strukturierten Durchführung von Softwareentwicklungsprojekten vorgestellt. Die drei hierbei eingeführten Arten zeigen dabei verschiedene Aspekte qualitativ hochwertiger Entwicklung auf. In den folgenden Abschnitten werden diese Lebenszyklusmodelle eine wichtige Rolle spielen. Zur besseren Vorstellbarkeit werden dabei allerdings die Kernelemente jedes Modells (Anforderungsanalyse, Spezifikation, Design, Implementierung, Testung und Auslieferung) betrachtet.

### 3 Betrachtungen einzelner Prozessbewertungsmodelle

Wie bereits angesprochen wurde in der Vergangenheit bereits eine Vielzahl verschiedener Prozessbewertungsstandards zur Bewertung und Verbesserung von Prozessen und Ressourcen innerhalb des Entwicklungsvorganges entwickelt.

In der Vielzahl der bereits in Abbildung 1 dargestellten Standards haben einige einen höheren Stellenwert erreicht. Drei von ihnen, die DIN EN ISO 9001:2000, CMM und CMMI gehören dabei wohl zu den weltweit am meisten genutzten Normen. Aufgrund dieses Umstandes soll es Gegenstand dieses Abschnittes sein, diese genauer zu untersuchen und deren Vorbeziehungsweise Nachteile aufzuzeigen. Weiterhin werden Gemeinsamkeiten und Unterschiede zwischen den dargestellten Standards einen Einblick in deren Zusammenhänge geben.

#### 3.1 ISO 9001:2000

Die ISO 9001:2000 ist ein neuer Standard für die Bewertung von Entwicklungsprozessen. Eingebettet in die ISO 9000er Reihe befasst er sich mit qualitätssteigernden Maßnahmen wie z.B. der Verstärkung der Kundenorientierung und löst damit die ISO 9000er Reihe aus dem Jahr 1994 ab. Die offensichtlichste Änderung in der neuen Fassung ist in der Neustrukturierung des Standards zu sehen. Statt der bisherigen zwanzig-elementigen Einteilung erscheinen in der aktuellen Version fünf Abschnitte. Obwohl alle Teile der ISO 9001:1994 (Darstellung nach [Wang2000])

- Verantwortung der Leitung
- Qualitätsmanagementsystem
- Vertragsprüfung
- Designlenkung
- Lenkung der Dokumente und Daten
- Beschaffung
- Lenkung der vom Kunden beigestellten Produkte
- Kennzeichnung und Rückverfolgbarkeit von Produkten
- Prozesslenkung
- Prüfungen
- Prüfmittelüberwachung
- Prüfstatus
- Lenkung fehlerhafter Produkte
- Korrektur- und Vorbeugungsmaßnahmen
- Handhabung, Lagerung, Verpackung, Konservierung und Versand
- Lenkung von Qualitätsaufzeichnungen
- Interne Qualitätsaudits
- Schulungen
- Wartung
- Statistische Methoden

in den fünf neuen Hauptabschnitten enthalten sind, lässt die ISO 9001:2000 zudem noch einen erheblich größeren Spielraum bei der Implementierung dieser Forderungen zu. Die Zuordnungsmöglichkeiten der einzelnen Abschnitte lassen sich durch die Betrachtung der Struktur des aktualisierten Standards erkennen:

- Qualitätsmanagementsystem
  - o Prozessdenken
  - o Dokumentationsanforderungen
- Verantwortung der Leitung
  - o Verpflichtung der Leitung
  - o Kundenorientierung
  - o Qualitätspolitik
  - o Planung
  - o Verantwortung, Befugnis und Kommunikation
  - o Managementbewertung
- Management von Ressourcen
  - o Bereitstellen von Ressourcen
  - o Personelle Ressourcen
  - o Infrastruktur
  - o Arbeitsumgebung
- Produktrealisierung
  - o Planung der Produktrealisierung
  - o Kundenbezogene Prozesse
  - o Entwicklung
  - o Beschaffung
  - o Produktion und Dienstleistungserbringung
  - o Lenkung von Überwachungs- und Messmitteln
- Messung, Analyse und Verbesserung
  - o Allgemeines
  - o Überwachung und Messung
  - o Lenkung fehlerhafter Produkte
  - o Datenanalyse
  - o Verbesserung

Hier lässt sich auch noch ein weiterer entscheidender Vorteil dieser Norm erkennen. Sie gibt strukturierte Vorgaben, die erfüllt werden müssen, um eine Qualitätssteigerung in den Produktionsprozessen erzielen zu können. Weiterhin stellen diese Konventionen die Basis für die Bewertung dieser Prozesse zur Verfügung.

Die Formulierung dieses Standards wurde von den Entwicklern sehr allgemein gehalten. Diese Eigenschaft kann sowohl positiv als auch negativ bewertet werden. Den positiven Aspekt stellt die Anwendbarkeit der Norm auf jede Form von Unternehmen dar. Auf der anderen Seite kann allerdings auch argumentiert werden, dass sie durch eben diese Freiheit zu

viel Freiraum und zu wenig konkrete Unterstützung für die Umsetzung der Anforderungen in spezielleren Entwicklungsbereichen wie beispielsweise der Softwareentwicklung bietet. Ein im Januar 2003 neu zur ISO 9000er Familie hinzugefügter Abschnitt mit der Bezeichnung ISO 9000-3 („Software Engineering – Guidelines for the Application of ISO 9001:2000 to Computer Software“) soll hier Abhilfe schaffen.

### 3.2 CMM

Laut [Maturity2003] stellt das Capability Maturity Model für Software, kurz SW-CMM, den bisher am häufigsten verwendeten Prozessbewertungsstandard dar. Diese Norm entstand durch eine Initiative des U.S. Verteidigungsministeriums mit dem Zweck der Effizienzermittlung von Software Vertragspartnern. Die Entwicklung dieses Modells begann in den 1980er Jahren am Software Engineering Institute (SEI) in Pittsburgh, Pennsylvania, USA. In der Zwischenzeit hat dieser Standard bereits auf der ganzen Welt Einzug gehalten und stellt somit einen nicht unerheblichen Standpfeiler zur Qualitätsbewertung von Softwareunternehmen dar.

Bei dem SW-CMM handelt es sich um ein Stufenmodell zur Prozessbewertung, in welchem 5 Reifegrade eine strukturierte Verbesserung der Prozess- und somit auch der Produktqualität ermöglichen. Die CMM-Stufen lassen sich anhand folgender Abbildung darstellen:

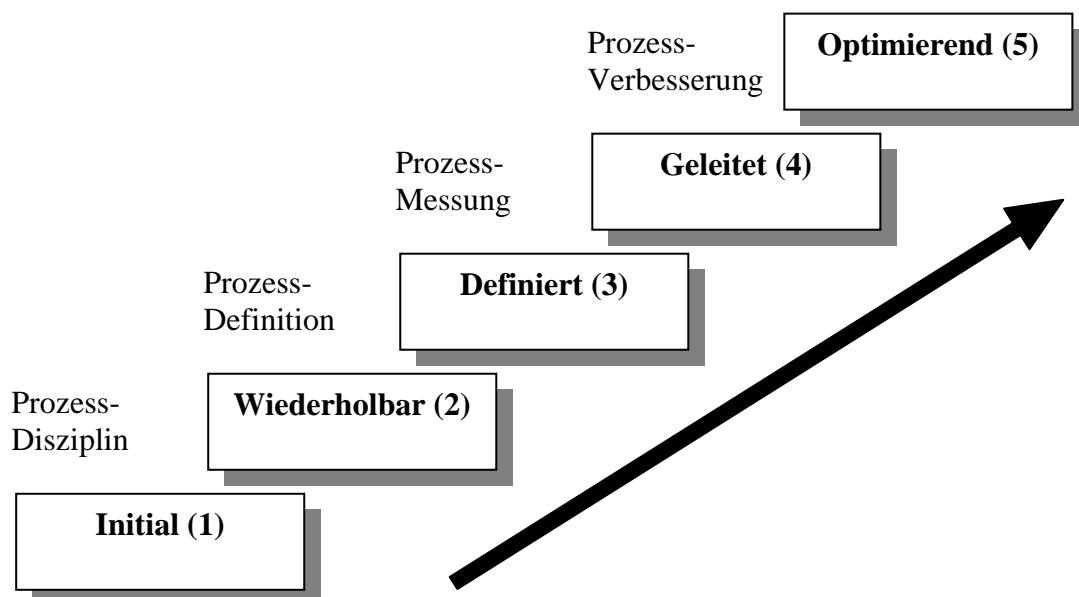


Abbildung 10: CMM-Stufen

Jede dieser Ebenen zeichnet sich durch spezielle Charakteristika (nach [Mutafelija2003]) aus:

- *Initial:* Der Softwareentwicklungsprozess läuft ad hoc und zeitweise auch unkoordiniert ab. Es existieren wenige bis keine definierten Prozesse und die Entwicklung ist abhängig von Einzelpersonen bzw. deren Fähigkeiten. Bei Verlust einer solchen Person geht zumeist auch das Wissen über den Stand der Entwicklung verloren. Diese Stufe stellt allerdings keine Stufe im eigentlichen Sinne, sondern nur den Initialzustand eines Unternehmens dar.
- *Wiederholbar:* In dieser Stufe wurden bereits die grundlegenden Projektmanagementprozesse zur Überwachung von Kosten, Planungen und Funktionalitäten erstellt und eingeführt. Die Einführung dieser Prozesse ermöglicht die Wiederholung von Erfolgen bei ähnlichen Aufgaben. Weiterhin ist hier eine effektive Kontrolle der Projektprozesse durch ein Projektmanagement zu verzeichnen, welches realistischen Planungen folgt, die auf vorhergehenden Projekten basieren.
- *Definiert:* Auf dieser Ebene sind die Prozesse für Management- und Entwicklungsaktivitäten dokumentiert, standardisiert und in den Softwareprozess der Organisation eingebettet. Weiterhin wird in jedem Projekt eine erprobte, angepasste Version dieses Softwareprozesses für die Entwicklung und Wartung der Software genutzt.
- *Geleitet:* Es werden detaillierte Messungen des Softwareprozesses und der Produktqualität durchgeführt und die Ergebnisse gesammelt und bewertet. Aufgrund dieser Prozesskontrolle ist es möglich Trends in der Prozess- bzw. Produktqualität vorherzusagen. Wenn diese Trends die festgelegten Schranken überschreiten, werden Korrekturmaßnahmen durchgeführt. Weiterhin ist zu bemerken, dass die entwickelte Software einen hohen Standard aufzeigt.
- *Optimierend:* In dieser Stufe wird eine kontinuierliche Prozessverbesserung durch eine Verbesserungen der einzelnen Prozesse und den koordinierten Einsatz innovativer Ideen bzw. neuer Technologien und Methoden ermöglicht.

Diese Charakteristika ermöglichen eine grobe Einschätzung der gegenwärtigen Prozessqualität innerhalb eines Softwareunternehmens, zeigen aber auf der anderen Seite bereits Anforderungen an Unternehmen auf, die entsprechende Ebenen erreichen wollen.

Eine feinere Strukturierung dieser Charakteristiken ist durch die Zuordnung von Unterebenen zu den einzelnen Stufen des Modells möglich. Diese Strukturelemente werden auch Keyprocess-Areas, Schlüsselprozessareale genannt. Die folgende Abbildung zeigt diese Zuordnung der einzelnen Schlüsselprozessareale zu den CMM-Ebenen:

- Initial:
  - o Keine Schlüsselprozesse, da Initialzustand
- Wiederholbar:
  - o Anforderungs-Management (AM)
  - o Software-Projektplanung (SPP)
  - o Software-Projektlenkung und –Verfolgung (SPLV)
  - o Software-Qualitätssicherung (SQS)
  - o Software-Konfigurations-Management (SKM)
  - o Software-Unterauftragnehmer-Management (SUM)
- Definiert
  - o Organisationsweiter Prozessfokus (OPF)
  - o Organisationsweite Prozessdefinition (OPD)
  - o Trainingsprogramm (TP)
  - o Integriertes Software-Management (ISM)
  - o Software-Produkt-Engineering (SPE)
  - o Gruppen-Koordination (GK)
  - o Peer-Review (PR)
- Geleitet
  - o Quantitatives Prozess-Management (QPM)
  - o Software-Qualitäts-Management (SQM)
- Optimierend
  - o Fehlervermeidung (FV)
  - o Technologie-Change-Management (TCM)
  - o Prozess-Change-Management (PCM)

Das Erfüllen der Anforderungen der Schlüsselprozessareale einer Ebene und aller untergeordneten Ebenen signalisiert das Erreichen des entsprechenden Levels. Daneben können diese Einzelprozesse auch bei der Bewertung des aktuellen Qualitätsstandes und bei Überlegungen bezüglich der weiterführenden Verbesserung der Prozessqualität verfeinernd Unterstützung bieten.

Die andauernde Weiterentwicklung der Standards ließ auch das SW-CMM nicht unberührt. Die Unterstützung für diese Norm wurde inzwischen eingestellt, da sich ein neuerer, das SW-CMM enthaltender Standard etabliert hat. Dieser heißt Capability Maturity Model Integration. Im folgenden Abschnitt soll dieser nun näher untersucht werden.

### 3.3 CMMI

Ebenfalls im Auftrag des US-Verteidigungsministeriums am SEI entwickelt, stellt das CMMI, das Capability Maturity Modell Integration, den Nachfolger von CMM dar. Wie bereits aus Abbildung 1 zu entnehmen ist, vereinigt CMMI unter Anderem verschiedene CMM-Arten, wie z.B. SW-CMM oder People-CMM, in einem einzigen Modell. Die Modelle, die in die Entwicklung des CMMI eingeflossen sind können in Abbildung 1 bereits erkannt werden. Einflüsse auf die Norm sind demnach in folgenden Modellen bzw. Normen zu finden:

- SW-CMM
- SA-CMM
- IPD-CMM
- EIA/IS 731
- PSM
- ISO 15939

Wie schon bei CMM existieren auch bei CMMI mehrere Versionen, die allerdings nicht wie bei CMM nur einzelne begrenzte Abschnitte beschreiben, sondern je nach Bedarf eines Unternehmens verschiedene Themenbereiche abdecken. Die einzelnen Versionen von CMMI sind folgende:

- CMMI-SE/SW → CMMI für Software- und Systementwicklung
- CMMI-SE/SW/IPPD → CMMI für Software- und Systementwicklung, sowie integrierte Prozess- und Produktentwicklung
- CMMI-SE/SW/IPPD/SS → CMMI für Software- und Systementwicklung, sowie integrierte Prozess- und Produktentwicklung und Softwarekauf

Ähnlich dem CMM weist jede Version dieses Modells auch eine stufenförmige Strukturierung auf. Zusätzlich dazu stellt es allerdings noch eine weitere Repräsentation zur Verfügung, die „kontinuierliche Darstellung“, welche eine feinere Analysierbarkeit der Grundsituation eines Unternehmens ermöglicht. Die Begründung für diese unterschiedlichen Darstellungen liegt laut [Ahern2003] in den verschiedenen Modellen, die in das CMMI eingeflossen sind. So zeigt beispielsweise das SW-CMM eine Stufenform auf, wohingegen ein weiterer Einflussfaktor, das Systems Engineering Capability Model ein kontinuierliches

Modell ist. Wiederum ein anderes enthaltenes Modell, das Integrated Product Development CMM, kurz IPD-CMM weist sowohl eine kontinuierliche als auch eine levelbasierte Darstellungsform auf. Die speziellen Eigenschaften dieser beiden CMMI-Repräsentationen sowie deren Vor- bzw. Nachteile sollen nun Gegenstand der Betrachtungen in den zwei folgenden Abschnitten sein.

### 3.3.1 CMMI – staged

Eine Entscheidung bezüglich einer Darstellungsform des CMMI ist auf den ersten Blick nicht sehr trivial. Überlegungen bezüglich der Anwendungsrichtung sind demzufolge notwendig. Laut [Ahern2003] ist die stufenförmige Repräsentation des CMMI in besonderem Maße nutzbar, wenn die Organisationsreife bei den Betrachtungen im Vordergrund steht.

Die Stufen-Version des CMMI ist ähnlich der CMM-Stufen. Sie umfasst ebenfalls 5 Stufen, welche durch folgende Darstellung näher geschildert werden.

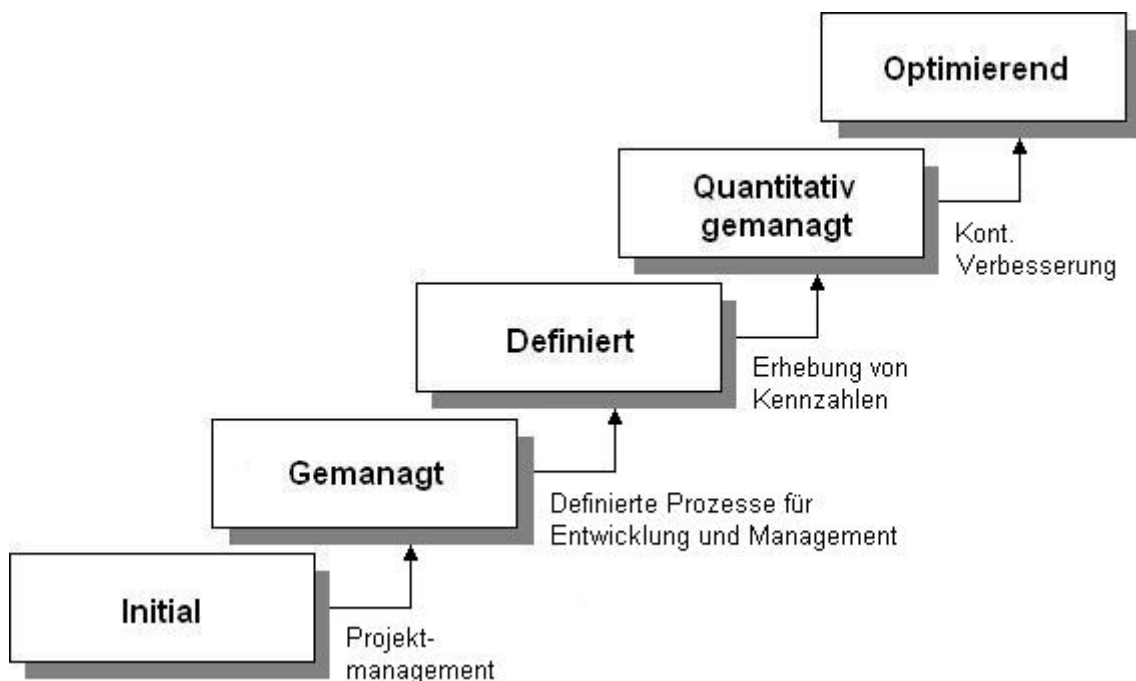


Abbildung 11: CMMI-Stufen

Ähnlich dem CMM werden auch beim CMMI jeder dieser Stufen Prozessgebiete zugeordnet, welche die Anforderungen der einzelnen Stufen grob charakterisieren. Diese Prozessareale lassen sich anhand folgender Tabelle (nach [Kneuper2003]) darstellen:

<b>Reifegrad</b>	<b>Prozessgebiete</b>
Initial	- keine -
Gemanagt	Anforderungsmanagement Projektplanung Projektverfolgung Management von Lieferantenvereinbarungen Messung und Analyse Qualitätssicherung Konfigurationsmanagement
Definiert	Anforderungsentwicklung Technische Umsetzung Produktintegration Verifikation Validation Organisationsweiter Prozessfokus Organisationsweite Prozessdefinition Organisationsweites Training Integriertes Projektmanagement Risikomanagement Entscheidungsanalyse und -findung
Quantitativ gemanagt	Performanz der organisationsweiten Prozesse Quantitatives Projektmanagement
Optimierend	Organisationsweite Innovation und Verbreitung Ursachenanalyse und Problemlösung

Tabelle 1: CMMI-Level mit zugehörigen Schlüsselprozessarealen

Diesen Prozessgebieten werden wiederum Ziele zugeordnet. Dabei unterscheidet man zwischen:

- Spezifischen und
- Generischen Zielen.

Die spezifischen Ziele gelten dabei jeweils nur für das entsprechende Prozessgebiet, wohingegen die generischen Ziele die Basis für eine dauerhafte und effiziente Umsetzung der spezifischen Ziele darstellen. So gibt es laut [Kneuper2003] auch nur zwei verschiedene generische Zielsetzungen:

- GG 2: Einen gemanagten Prozess institutionalisieren
- GG 3: Einen definierten Prozess institutionalisieren

Das erste generische Ziel ist dabei der Stufe 2 und das zweite den Stufen 3-5 zugeordnet.

Zum Erreichen der jeweiligen Ziele beinhaltet der CMMI-Text bereits sowohl spezifische als auch generische Praktiken. Eine komplette Darstellung der Zusammenhänge kann anhand folgender Abbildung nach [Ahern2003] gegeben werden:

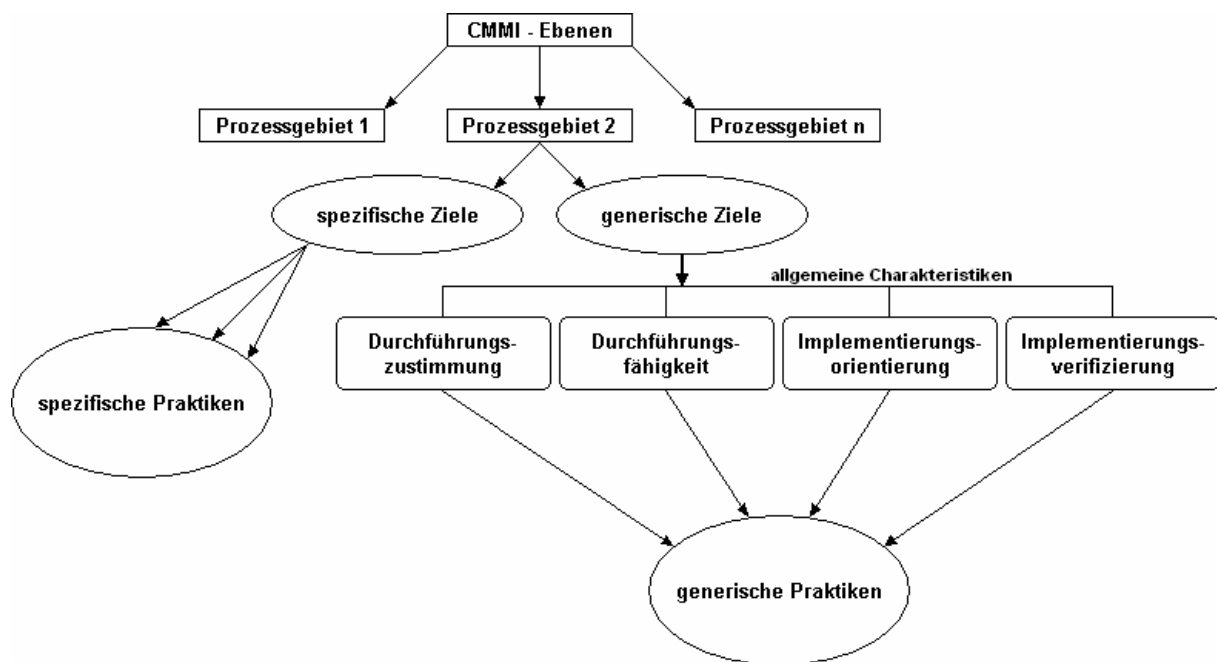


Abbildung 12: Struktur der CMMI-Stufenform

Laut [Ahern2003] bezieht sich die stufenförmige Anordnung auf den Schlüsselbegriff „Reife“ (maturity). Ein Vorteil dieser Repräsentation ist demzufolge die Möglichkeit der Bestimmung eines organisationsübergreifenden Reifegrades. Weiterhin ist es möglich, eine strukturierte Verbesserung des gesamten Unternehmens durchzuführen, da die einzelnen Prozessareale eindeutig zu den CMMI-Leveln zugeordnet sind.

Aus diesen Zusammenfassungen zu Ebenen resultiert die Unfähigkeit dieser Erscheinungsform, Verbesserungen einzelner Prozessareale in ausreichendem Maß zu unterstützen. Eine Nutzung der kontinuierlichen Darstellung kann demzufolge ebenso sinnvoll sein, wie die Nutzung der Stufen-Repräsentation.

### 3.3.2 CMMI – continuous

Laut [Ahern2003] ist der Hauptbezugspunkt der kontinuierlichen Darstellung die Ausrichtung auf den Begriff „Fähigkeit“ (capability). Die Unterstützung, die durch diese Repräsentation gegeben werden soll, bezieht sich demzufolge auf eine individuell, auf die einzelnen Areale ausgerichtete, schrittweise Evolution beziehungsweise Messung der jeweils erzielten Verbesserungsergebnisse.

Die kontinuierliche Darstellung des CMMI – Modells ermöglicht weiterhin eine feinere Darstellung der Prozessqualität einzelner Prozesse innerhalb einer Organisation. Dazu besitzt sie eine Zuordnung von „Fähigkeitsgraden“. Diese gliedern sich nicht wie bei der Stufenform in 5 Ebenen, sondern 6 Fähigkeitsgrade. Diese Kategorien sind:

- 0 – unvollständig
- 1 – durchgeführt
- 2 – gemanagt
- 3 – definiert
- 4 – quantitativ gemanagt
- 5 - optimierend

Bei dieser Form des CMMI wird nicht wie in der Stufenform eine Bewertung der Prozesse anhand der erreichten Prozessareale einer Ebene durchgeführt, sondern jeder Schlüsselprozess getrennt betrachtet. Dazu werden hier diese Prozessgebiete nicht einer Stufe, sondern entsprechend [Mutafelija2003] Kategorien zugeordnet:

Kategorie	Prozessgebiet
Prozessmanagement	Organisationsweiter Prozessfokus Organisationsweite Prozessdefinition Organisationsweites Training Performanz d. organisationsweiten Prozesse Organisationsweite Innovation und Verbreitung

Projektmanagement	Projektplanung Projektverfolgung und -Steuerung Management von Lieferantenvereinbarungen Integriertes Projektmanagement Risikomanagement Quantitatives Projektmanagement
Entwicklung	Anforderungsmanagement Anforderungsentwicklung Technische Umsetzung Produktintegration Verifikation Validation
Unterstützung	Konfigurationsmanagement Qualitätssicherung von Prozessen und Produkten Messung und Analyse Ursachenanalyse und Problemlösung Entscheidungsanalyse und –Findung

Tabelle 2: Zuordnung der Schlüsselprozessareale zu CMMI-Kategorien

Die feingliederige Aufschlüsselung, welche sich durch die Einzelbewertung der Prozessgebiete ergibt, zieht sowohl Vor- als auch Nachteile nach sich. Der größte Vorteil ist die Möglichkeit, sich nur auf bestimmte, auf die Tätigkeitsbereiche eines Unternehmens zugeschnittene, Prozessareale zu konzentrieren. Das bedeutet, ein Unternehmen, welches sich hauptsächlich mit dem Support beschäftigt, kann beispielsweise in „Messung und Analyse“, „Entscheidungsanalyse und –findung“ und „Ursachenanalyse und Problemlösung“ einen hohen Fähigkeitsgrad erreichen, während Engineering-Areale wie „Anforderungsmanagement“ oder „Produktintegration“ nur in geringerem Maße behandelt werden müssen.

Darin liegt aber gleichzeitig auch der größte Nachteil dieser Repräsentation. Die Prozesse eines Unternehmens, welches nach dem kontinuierlichen Verfahren arbeitet, sind aufgrund ihrer mitunter extrem unterschiedlichen Fähigkeitsgrade schwer zu bewerten.

Ein weiterer Unterschied zur CMMI-Stufendarstellung ist die Zuordnung von fünf verschiedenen generischen Zielen zu den einzelnen Fähigkeitsgraden. In der stufenförmigen

Repräsentation handelt es sich um lediglich vier generische Ziele, entsprechend den vier Ebenen, die eine Erfüllung dieser Ziele erwarten. Hinzu kommt hier noch der Fähigkeitsgrad null, in welchem keines der generischen Ziele erfüllt ist. Diese Ziele sind in folgender Darstellung nach [Kneuper2003] ersichtlich:

- GG 1: Spezifische Ziele erreichen
- GG 2: Einen gemanagten Prozess institutionalisieren
- GG 3: Einen definierten Prozess institutionalisieren
- GG 4: Einen quantitativ gemanagten Prozess institutionalisieren
- GG 5: Einen optimierenden Prozess institutionalisieren

Zum Erreichen der einzelnen Ziele sind wie auch schon bei der stufenförmigen Darstellung im CMMI-Text unterstützende Praktiken (spezielle und generische) definiert.

### 3.3.3 Überführung

Die unterschiedliche Einteilung beider Repräsentationen in einerseits 5 Reifegrade bei der Stufenform und andererseits 6 Fähigkeitsgrade in der kontinuierlichen Darstellung erschwert eine Überführung der verschiedenen Sichten ineinander. Da aber auch innerhalb eines Unternehmens, welches eine Prozessverbesserung nach der kontinuierlichen Darstellung verwendet eventuell die organisationsweite Prozessreife bestimmt werden soll, ist dieser Umstand sehr hinderlich. Deshalb beschreibt das CMMI, zur Einhaltung des ursprünglichen Gedankens eines Modells mit zwei verschiedenen Repräsentationen, auch wie beide Darstellungen ineinander überführt werden können. Die folgenden Abbildungen sind [Ahern2003] entnommen und zeigen diese Zusammenhänge beider Modellierungen.

	FG1	FG2	FG3	FG4	FG5
Maturity Level 2 Prozess-Areale	Zielprofil 2				
Maturity Level 3 Prozess-Areale					
Maturity Level 4 Prozess-Areale					
Maturity Level 5 Prozess-Areale					

Tabelle 3: Benötigter Fähigkeitsgrad für Level 2

	FG1	FG2	FG3	FG4	FG5
Maturity Level 2 Prozess-Areale	Zielprofil 3				
Maturity Level 3 Prozess-Areale					
Maturity Level 4 Prozess-Areale					
Maturity Level 5 Prozess-Areale					

Tabelle 4: Benötigter Fähigkeitsgrad für Level 3

	FG1	FG2	FG3	FG4	FG5
Maturity Level 2 Prozess-Areale	Zielprofil 4				
Maturity Level 3 Prozess-Areale					
Maturity Level 4 Prozess-Areale					
Maturity Level 5 Prozess-Areale					

Tabelle 5: Benötigter Fähigkeitsgrad für Level 4

	FG1	FG2	FG3	FG4	FG5
Maturity Level 2 Prozess-Areale	Zielprofil 5				
Maturity Level 3 Prozess-Areale					
Maturity Level 4 Prozess-Areale					
Maturity Level 5 Prozess-Areale					

Tabelle 6: Benötigter Fähigkeitsgrad für Level 5

Dieser Überführungsmechanismus zeigt bereits in ausreichendem Maß die Konsistenz beider Darstellungsformen.

### **3.4 Unterschiede / Gemeinsamkeiten der einzelnen Modelle**

Nachdem nun drei der wohl wichtigsten Normen näher erläutert wurden, stellt sich die Frage nach deren Vor- beziehungsweise Nachteilen. Daraus resultierend schließt sich die Frage an, welches der vorgestellten Modelle das Beste ist.

Die Beantwortung dieser Frage ist allerdings nicht sehr einfach, da die einzelnen Standards sich in ihrer Ausrichtung und weiteren Eigenschaften sehr unterscheiden. Es erscheint also

sinnvoll, sich die Vor- und Nachteile der einzelnen Normen zu veranschaulichen und daraus resultierend das für die eigenen Wünsche am besten geeignete Modell auszuwählen.

Als erstes kann man festhalten, dass SW-CMM und CMMI in einem engen Zusammenhang stehen, da das SW-CMM, neben dem EIA/IS 731 (das Capability Modell für Softwareentwicklung der Electronics Industrie Alliance) und dem IPD-CMM (CMM für integrierte Produktentwicklung) einen der Grundbausteine des CMMI bildet. Eine weitere Berücksichtigung des SW-CMM erübrigt sich deshalb. Ein weiterer Grund hierfür ist die Einstellung der Unterstützung für CMM. Aufgrund dieser Fakten ist eine Überprüfung der Zusammenhänge zwischen CMM und der DIN EN ISO 9001:2000 überflüssig. Stattdessen rückt eine Betrachtung der Vor- beziehungsweise Nachteile von CMMI gegenüber der ISO 9001 in den Vordergrund.

Der Vergleich von ISO 9001:2000 und CMMI gestaltet sich weitaus schwieriger, da beide in ähnlichem Maße Vor- bzw. Nachteile aufweisen.

Der Hauptunterschied zwischen beiden Standards ist die Zielgruppe. Während die DIN EN ISO 9001:2000 aufgrund ihrer sehr allgemein gehaltenen Form auf jeden Industriezweig anwendbar ist, bezieht sich das CMMI lediglich auf die Entwicklung von Software. Dieser allgemeine Stil der ISO allerdings auch einer ihrer Nachteile. Durch diese Form ist es schwierig die Forderungen des Standards auf die Softwareentwicklung zu übertragen. Dieser Umstand wird durch die mangelnde Unterstützung bezüglich der Anwendung der Norm noch weiter verschlechtert. Abhilfe soll hier ein neu zur ISO 9000er Familie hinzugefügter Standard, die ISO 9000-3, schaffen. Diese Norm gibt allerdings ebenfalls keine konkreten Hinweise oder Praktiken für die Umsetzung der ISO-Anforderungen vor, sondern zeigt nur, wie der Normtext aus der ISO 9001 bezogen auf Softwareunternehmen zu verstehen ist.

An dieser Stelle zeigen sich die Vorzüge des CMMI. Aufgrund seiner Spezialisierung auf Softwareentwicklungsprozesse ist eine allgemeine Formulierung nicht notwendig. Weiterhin ermöglicht die Ausrichtung auf einen einzelnen Industriezweig eine bessere Unterstützung für die konkrete Anwendung des Standards. Im Fall des CMMI wird die Anwendung ermöglicht, indem spezielle Ziele für das Erreichen einer Reifeebene definiert werden. Ferner befähigt CMMI den Anwender zur Umsetzung dieser Anforderungen, indem es Methoden vorschlägt, die genutzt werden können, aber nicht zwingend umzusetzen sind. Hierbei werden zwei verschiedene Typen unterschieden. Zum einen die spezifischen, einem Areal eindeutig zugeordneten und zum anderen die generischen, in mehreren Arealen wiederkehrenden Ziele

beziehungsweise Praktiken. Diese Eigenschaften machen CMMI zu einem mächtigen Werkzeug bei der Prozessverbesserung. Die Allgemeinen Unterschiede zwischen CMMI und ISO 9001 sind in folgender Tabelle entsprechend [CMMI01] nochmals dargestellt.

	<b>CMMI-Assessments</b>	<b>DIN EN ISO 9001:2000</b>
<b>Gegenstand</b>	Für reine Software-Entwicklungsprozesse vorgesehen	Vielzahl industrieller Organisationen, Produkte und Abläufe
<b>Ziel</b>	Detaillierte Ziel- und Prioritätsvorgaben zur Verbesserung des Prozesses	Nachweis der Qualifikation zur Erzeugung qualitätsgerechter Resultate
<b>Status</b>	Nützliches Hilfsmittel zur Problemanalyse und Prozessverbesserung	Fester Industriestandard
<b>Forderungen</b>	Hierarchie von Forderungen in Abhängigkeit der Stufen	Minimalanforderungen (ausnahmslos zu erfüllen)
<b>Basis</b>	Flexibles Modell	Starrer Normtext
<b>Ergebnis</b>	Ist-Stand, Stärken- und Schwächen-Profil	Anerkanntes Zertifikat
<b>Kosten/Nutzen</b>	Einsparungen durch Prozessverbesserung vs. Kosten für Assessments und Einführung der Verbesserungen	Nutzen ist durch das erteilte Zertifikat begründet

Tabelle 7: Vergleich von CMMI und DIN EN ISO 9001:2000

### **3.5 Zusammenfassung**

Die Absicht dieses Abschnitts war eine Einführung der wichtigsten Prozessbewertungs-Standards. Grundlage der weiterführenden Untersuchungen wird allerdings das Capability Maturity Model Integration sein. Ein Grund für diese Entscheidung ist die Nutzungshäufigkeit. Da die Unterstützung für CMM, wie bereits erwähnt nicht mehr weitergeführt wird, ist eine Anpassung der Unternehmen mit CMM-Ausrichtung auf CMMI sehr wahrscheinlich, was CMMI zu einem der meistgenutzten Modelle macht.

Ein weiterer positiver Faktor für den Einsatz von CMMI gegenüber der ISO 9001:2000 ist die Stufenform. Diese Granularität ermöglicht eine strukturierte Verbesserung der Entwicklungsprozesse. Die ISO 9001:2000 hingegen verlangt das Erfüllen aller Anforderungen zum Erreichen einer Zertifizierung. Durch diese Eigenschaft, kann es zu unkoordinierten Entwicklungen bei der Qualitätsverbesserung kommen, was durch die levelbasierte Ausrichtung des CMMI eingeschränkt werden kann.

Einen der Hauptgründe für die weitere Verwendung von CMMI im Rahmen dieser Arbeit stellt allerdings seine Ausrichtung auf die Betrachtung software-relevanter Systeme dar. Das wird noch verstärkt durch die Überlegungen im Zusammenhang mit dem praktischen Ansatz

des abschließenden Kapitels, da es sich dabei um Betrachtung bezüglich eines Softwareunternehmens handelt.

## 4 Einführung in Size-Measurement

Bereit in Kapitel „2.2 Software-Engineering und Softwareprozessbewertung“ wurden die Qualitätsmerkmale von Software aufgezeigt. In diesem Zusammenhang ist durch die Softwaregröße oder auch Size eine wesentliche, viele dieser Merkmale beeinflussende Eigenschaft gegeben. Beispielsweise ist für ein Programm mit großem Umfang ein höherer Wartungsaufwand zu erwarten aber auch die Prüfbarkeit, die Verständlichkeit oder das Zeitverhalten werden sicherlich negativ beeinflusst. Dieser Abschnitt soll sich nun mit allgemeinen und speziellen Betrachtungen zur Umfangsbestimmung von Software auseinandersetzen.

### 4.1 Allgemeine Betrachtungen

Aus den gezeigten Gründen ist es notwendig, Kennzahlen von Software zu bestimmen, die eine Bewertung des Umfangs zulassen. Da Software an sich allerdings keine derartigen Kennzahlen aufweist, ist es notwendig, die Eigenschaften von Software auf eine geeignete Art und Weise auf Zahlen abzubilden. Konkret geschieht dies durch den Einsatz von Metriken, oder in unserem Fall von Softwaremetriken (siehe [Wiki1]):

Eine Software (qualitäts-)metrik ist eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet. Dieser berechnete Wert ist interpretierbar als der Erfüllungsgrad einer Qualitätseigenschaft.

Das bedeutet, dass eine Softwaremetrik uns die Möglichkeit bietet, Eigenschaften von Software auf Zahlen abzubilden, was diese wiederum für uns vergleichbar bzw. verwertbar macht. In unserem speziellen Fall bildet also eine Metrik Softwareeigenschaften auf Zahlenwerte ab.

Die Eigenschaften von Software sind allerdings ebenso wie die Softwaregröße schwierig vorstellbar. Nichtsdestotrotz existiert eine unglaublich große Menge Softwareeigenschaften. Dabei unterscheidet [Fenton1997] vier verschiedene Attribute bei der Größe von Software.

- 1) Physikalische Größe
- 2) Komplexität
- 3) Funktionale Größe

#### 4) Reuse

Die physikalische Größe stellt hierbei den Umfang des Produktes, speziell des Codes dar. Die Funktionalität beschreibt den durch das Produkt zur Verfügung gestellten Funktionsumfang. Reuse stellt die Wiederverwendung von Codeteilen oder Produkten dar, was bei wirtschaftlich ausgerichteten Überprüfungen eine größere Rolle spielen könnte. Bei der Komplexität kann man, ebenfalls nach [Fenton1997] folgende Arten unterscheiden:

- Problemkomplexität
  - o Komplexität eines zugrunde liegenden Problems
- Algorithmische Komplexität
  - o Komplexität des Algorithmus der das Problem löst
- Strukturelle Komplexität
  - o Komplexität des Codes der Software
- Kognitive Komplexität
  - o Beschreibt den Aufwand zum Verstehen des Softwareproduktes

Im Rahmen dieser Arbeit werden die verschiedenen Komplexitätsarten jedoch nur erwähnt und nicht weiter berücksichtigt. Für nähere Betrachtungen sei dabei auf [Fenton1997] verwiesen, da dort tiefer gehende Betrachtungen zu den einzelnen Komplexitätsarten durchgeführt werden. Die Eigenschaften von Software, die in dieser Ausarbeitung im Vordergrund stehen, sind die physikalische und die funktionale Größe.

## **4.2 Physikalische Größe**

### **4.2.1 Lines of Code**

Das wohl älteste Maß zur Bestimmung der physikalischen Größe und somit häufig genutzte Basis zur Ermittlung der strukturellen Komplexität sind die Lines of Code. Dieses Maß bestimmt, wie der Name bereits vermuten lässt die Anzahl der Codezeilen. Allerdings ist die Menge der programmierten Zeilen an sich ein wenig aussagekräftiges Maß, da jeder Programmierer einen eigenen Stil besitzt, und jede Programmiersprache andere Voraussetzungen stellt. So kann ein Programmierer innerhalb einer Zeile mehrere Aktionen durchführen, wobei ein anderer eine einzige Aktion über mehrere Zeilen ausdehnt. Eine genaue Definition, wie die LOC bewertet werden sollen und Überlegungen, welche Schlüsse man sich aus der Messung erhofft, sind also im Vorfeld dieser Messung unumgänglich.

Wichtig ist ebenfalls, sich in diesem Zusammenhang bewusst zu machen, dass ein Programm mit einer großen Menge an Codezeilen nicht in jedem Fall eine höhere strukturelle Komplexität besitzen muss als ein Programm mit wenigen Zeilen.

Eine Liste möglicher LOC-Variationen und deren Interpretationen werden in der folgenden Abbildung nach [DumkeSE2] gegeben:

LOC-Variante	Interpretation
Anzahl aller Programmzeilen	Gesamter Umfang
Anzahl abarbeitbarer Zeilen	Algorithmenimplementationsgröße
Anzahl geänderter Zeilen	Änderungshäufigkeit
Anzahl wiederverwendeter Zeilen	Reuse-Anteil
Anzahl Kommentarzeilen	Kommentaranteil

Tabelle 8: LOC-Varianten

#### 4.2.2 Halstead

Eine weitere Möglichkeit für die an den physischen Eigenschaften des Codes festgemachte Umfangsbestimmung sind die 1977 von Halstead veröffentlichten Softwaremaße. Sie basieren laut [Neumann2003] auf vier grundlegenden Software-Eigenschaften:

- $n_1$  – Anzahl unterschiedlicher Operatoren
- $n_2$  – Anzahl unterschiedlicher Operanden
- $N_1$  – Anzahl aller benutzten Operatoren
- $N_2$  – Anzahl aller benutzten Operanden

Dabei stellen Operanden Bezeichner (Variablen) dar und Operatoren sind Aktionen beschreibende Symbole (+, -, for, Funktionen, etc). Auf der Basis dieser Eigenschaften definierte Halstead nun seine Softwaremaße. Einige wichtige dieser Maße sind folgende:

- Vokabular:
  - o Formel:  $n = n_1 + n_2$
  - o Bedeutung: Zeigt die Gesamtanzahl der verwendeten Bezeichner, unabhängig von der Anzahl ihres Vorkommens
- Implementationslänge:

- Formel:  $N = N_1 + N_2$
- Bedeutung: Anzahl aller Operatoren und Operanden. Zeigt dadurch einige Gemeinsamkeiten mit LOC
- Volumen:
  - Formel:  $V = (N_1 + N_2) * \log_2 (n_1 + n_2) = N * \log_2 n$
  - Bedeutung: Abschätzung der finalen Programmgröße, wenn eine Vielzahl der benötigten Operanden/Operatoren bereits bekannt sind
- Schwierigkeit:
  - Formel:  $D = \frac{(n_1 * N_2)}{(2 * n_2)}$
  - Bedeutung: Abschätzung der Schwierigkeit ein Programm zu schreiben; Somit auch verwendbar für die Komplexität; Wird größer bei Anstieg der Operatorenanzahl oder Operandennutzung
- Zeitlicher Aufwand:
  - Formel:  $E = \frac{(n_1 * N_2 * (N_1 + N_2) * \log_2 (n_1 + n_2))}{(2 * n_2 * S)} = D * V$
  - Bedeutung: Abschätzung des zeitlichen Aufwands für das Schreiben eines Programms basierend auf S (Menge der Unterscheidungen beim Programmieren eines Menschen;  $5 < S < 20$ )

Ein Nachteil aller hier aufgezeigten Metriken ist, dass sie nur für den Einsatz bei der Komplexitätsbestimmung von Software-Modulen konzipiert worden sind. Eine Anwendung auf komplette Softwareprodukte ist dabei wenig sinnvoll. Weiterhin ist nachteilig, dass die zur Bestimmung der einzelnen Maße notwendigen Daten, zum Beispiel die Anzahl der Operatoren und Operanden in der Halstead-Metrik, bei objektorientierten Sprachen nur schwer zu bestimmen sind. Gründe hierfür sind die speziellen Konzepte, die in diesen Sprachen Anwendung finden, wie beispielsweise Vererbung oder Abstraktion.

Zur Umgehung dieser Nachteile erscheint eine weitere Möglichkeit zur Bestimmung der Größe von Software unumgänglich. Ein solcher Ansatz soll im nun folgenden Abschnitt in den Vordergrund rücken.

### **4.3 Funktionale Größe von Software**

Nach dieser kleinen Übersicht über Ansätze zur Bestimmung der physikalischen Größe von Software wenden wir uns nun einem anderen Ansatz zur Bestimmung der Softwaregröße zu. Die funktionale Größenmessung ist ein Feld der Umfangsbestimmung von Software, welches

in der heutigen Zeit immer weiter an Bedeutung gewinnt. Dies ist nicht zuletzt darauf zurückzuführen, dass sie Softwareentwicklungsmethoden und –Prinzipien zu den im Bereich der Softwareentwicklung bereits vorhandenen hinzufügt.

Die funktionalen Größenmessung wurden im Jahr 1979 mit der Entwicklung der „Function Point Analysis“ durch Allan J. Albrecht eingeführt. In den folgenden Jahren wurde seine Idee vielfach weiterentwickelt und ausgebaut. Die folgende Graphik (basierend auf [Geocity1] und [Lother1]) gibt einen Überblick über die verschiedenen Formen der funktionalen Größenmessung, die alternativ zu und basierend auf Albrechts Ansatz, im Laufe der Zeit entwickelt wurden:

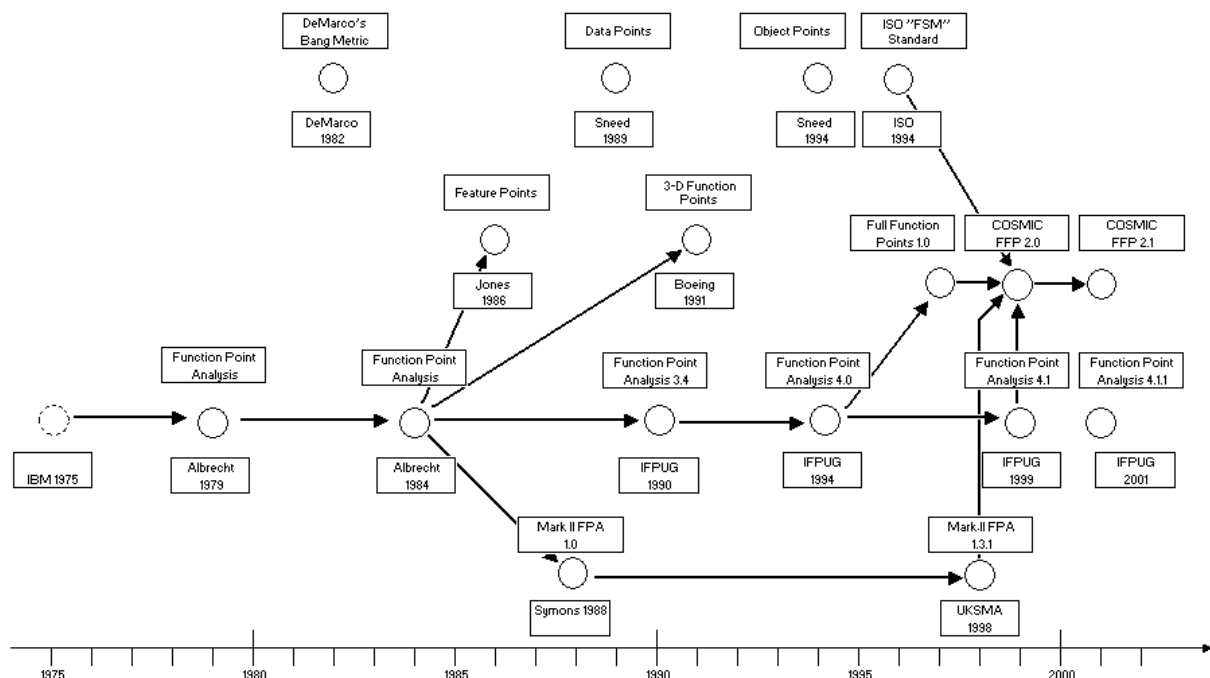


Abbildung 13: Entwicklung der funktionalen Größenmessung

Der Grund für die Entwicklung der funktionalen Größenmessung war die Schaffung eines einheitlichen Mechanismus zur Definition von funktionalen Anforderungen sowohl für Entwickler als auch für Nutzer. In diesem Zusammenhang wurde festgestellt, dass es für das Verständnis der Nutzeranforderungen notwendig ist, die Interaktion des Nutzers mit dem System zur Basis zu nehmen. Aufgrund dieser Feststellungen basieren die Analysen bei dem „functional size measurement“ auf den verschiedenen Interaktionsarten eines Nutzers mit einem System. Die im Allgemeinen bei der funktionalen Größenmessung berücksichtigten Parameter sind in der folgenden Abbildung als Pfeile dargestellt:

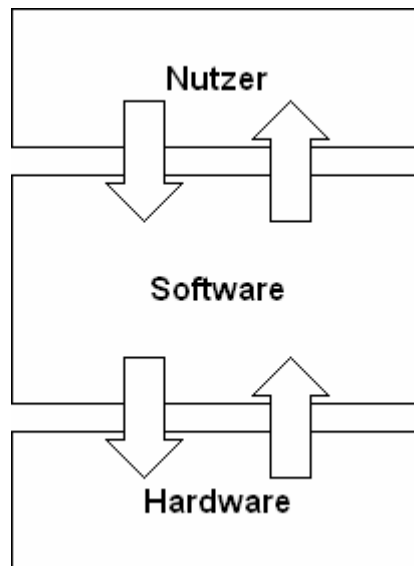


Abbildung 14: Parameter der funktionalen Größenmessung

Die Art der zur Analyse herangezogenen Daten differiert in den verschiedenen Modellen. Im folgenden Abschnitt dieses Kapitels sollen nun ausgehend von Albrechts Ansatz einige der grundlegenden Konzepte der funktionalen Größenmessung betrachtet werden.

#### 4.3.1 Albrechts Ansatz

Albrecht nutzte in seiner Herangehensweise sogenannte „function points“. Diese sollen die in den Spezifikationen beschriebene Menge an Funktionalität eines Systems darstellen. Dies sollten nach Albrechts Ansicht technologie-unabhängig sein. Zur Bestimmung der Funktionalität wird dabei nach [Fenton1997] zuerst die Menge an:

- Externen Eingaben
- Externen Ausgaben
- Externen Anfragen
- Externen Dateien
- Internen Dateien

bestimmt. Anschließend wird jede der bestimmten Einheiten bewertet indem sie als „trivial“, „durchschnittlich“ oder „komplex“ eingestuft und danach entsprechend folgender Tabelle nach [Fenton1997] gewichtet wird.

Einheit	trivial	durchschnittlich	Komplex
Externe Eingaben	3	4	6
Externe Ausgaben	4	5	7
Externe Anfragen	3	4	6
Externe Dateien	7	10	15
Interne Dateien	5	7	10

Tabelle 9: Wichtungstabelle für UFC-Bestimmung

Aus den so gewonnenen gewichteten Einheiten werden anschließend die „unangepassten Funktionspunkte“ mittels folgender Formel bestimmt:

$$UFC = \sum_{i=1}^{15} ((\text{Anzahl der Einheiten vom Typ } i) * \text{Wichtung } i)$$

Zur finalen Berechnung der Funktionspunkte muss im nächsten Schritt ein „technischer Komplexitätsfaktor“ bestimmt werden. Dieser wird ermittelt, indem die Faktoren:

F <sub>1</sub> : Verlässliches Back-Up und Wiederherstellung
F <sub>2</sub> : Daten Kommunikationen
F <sub>3</sub> : Verteilte Funktionen
F <sub>4</sub> : Performanz
F <sub>5</sub> : Stark belastete Konfiguration
F <sub>6</sub> : Online Dateneinträge
F <sub>7</sub> : Operationelle Einfachheit
F <sub>8</sub> : Online Update
F <sub>9</sub> : Komplexes Interface
F <sub>10</sub> : Komplexe Bearbeitung
F <sub>11</sub> : Wiederverwendbarkeit
F <sub>12</sub> : Einfache Installation
F <sub>13</sub> : Vielfache Seiten
F <sub>14</sub> : Erleichterte Veränderungen

Tabelle 10: Kostentreiber für technischen Komplexitätsfaktor

entsprechend ihrer Wichtigkeit in Klassen eingeteilt und entsprechend gewichtet werden. Diese Wichtung geschieht entsprechend der in folgender Tabelle nach [Rook1991] dargestellten Einstufung der einzelnen Faktoren:

- 0 Faktor hat keinen Einfluss oder ist unbekannt
- 1 Unsignifikanter Einflussfaktor
- 2 Moderater Einflussfaktor
- 3 Faktor ist durchschnittlich wichtig
- 4 Signifikanter Einflussfaktor
- 5 Faktor ist essentiell wichtig

Tabelle 11: Wichtungstabelle für techn. Komplexitätsfaktor

und wird anschließend mittels folgender Formel zum technischen Komplexitätsfaktor zusammengefasst:

$$UCF = 0.65 + 0.01 \sum_{i=1}^{14} F_i$$

Die Funktionspunkte werden nun abschließend mit Hilfe der Formel:

$$FP = UCF * TCF$$

berechnet. Mit diesen Funktionspunkten lassen sich nun Abschätzungen durchführen. Beispielsweise kann, wenn Wissen über den Zeitbedarf pro Entwickler für die Entwicklung eines Funktionspunkts vorliegt, die Entwicklungszeit abgeschätzt werden. Daraus wiederum können die Kosten für personelle Ressourcen abgeschätzt werden. So stellen die Funktionspunkte eine entscheidende Basis für die Abschätzung vieler Softwareeigenschaften dar.

#### **4.3.2 Full Function Points**

Im Laufe der Zeit hat Albrechts Ansatz immer wieder Verbesserungen erfahren. So wurden die Zählparameter angepasst und andere Einheiten der Software traten mehr in den Vordergrund. Eine der fortschrittlichsten Varianten auf der Basis des originalen Ansatzes ist COSMIC Full Function Points v. 2.1 aus dem Jahr 2001. Nach [Lothar1] handelt es sich hier um eine Version, die den Anspruch erhebt, neben MIS Software sowohl Realtime-, System- als auch technische Software abzudecken. Die Bestimmung basiert hierbei auf der Annahme, dass Datenbewegungen die Größe eines Systems bestimmen. Bei der COSMIC FFP – Methode werden aufgrund dieser Annahme nunmehr folgende Einheiten genutzt:

- Eingänge
- Ausgänge
- Lesevorgänge
- Schreibvorgänge

Im Zusammenhang mit einem eingebetteten Layer – Prinzip (Darstellung nach [Lothar1]):

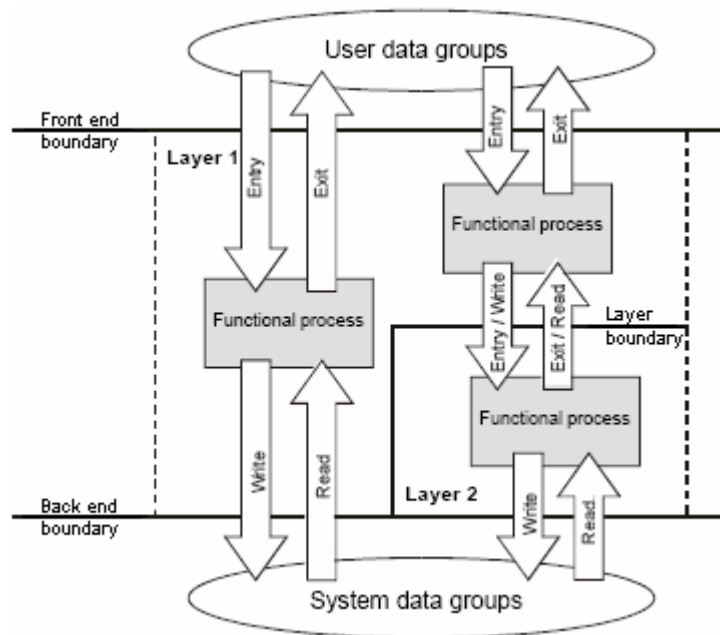


Abbildung 15: Layer-Prinzip von COSMIC FFP

ergibt sich daraus ein mächtiges Werkzeug zur Funktionalitätsbestimmung von Software-Produkten.

### 4.3.3 Weitere Ansätze

Neben der Bestimmung von Functions Point wurden allerdings noch weitere Ansätze zur Bestimmung der Funktionalitäten eines Softwareproduktes verfolgt. Die grundlegende Gemeinsamkeit aller Ansätze ist jedoch der Versuch spezielle, festgelegte Eigenschaften der Software auf dem Nutzer zur Verfügung gestellte Funktionalitäten umzuschlagen. Andere Herangehensweisen sind beispielsweise:

- DeMarcos Bang Metric
  - o Ansatz für Software Systeme und Wissenschaftssoftware
- Objekt Points
  - o Ansatz für objektorientierte Software
- Data Points
  - o Ansatz die Messbasis von Funktionen auf deren Datenrepräsentation zu verschieben
- Feature Points
  - o Ansatz für Software Systeme und Realtime – Anwendungen

#### **4.4 Zusammenfassung**

Die Größe ist eine der wichtigsten Eigenschaften von Software. Für die Bestimmung dieser Eigenschaft existiert eine Vielzahl von Herangehensweisen. Die funktionale Größenmessung ist neben der Ermittlung der physikalischen Größe über Lines of Code oder Halstead, eine dieser Praktiken. Die einzige bisher angesprochene positive Eigenschaft der funktionalen Größenmessung gegenüber den anderen Ansätzen ist die Technologieunabhängigkeit. Sie stellt einen der größten Vorteile dieser Herangehensweise gegenüber anderen Metriken dar, da sich aufgrund dieser Eigenschaft die Softwaregröße bereits in einem frühen Stadium der Entwicklung bestimmen lässt. Eine weitere Eigenschaft ist die Nutzersicht die durch funktionale Größenmessung gegeben wird. Diese beruht auf der Bestimmung der funktionalen Größe über die an das Produkt gestellten Nutzer-Anforderungen.

Aufgrund dieser Eigenschaften wird die funktionale Größenmessung einen der Kerngedanken in den folgenden Betrachtungen darstellen. Da jede der vorgestellten Methoden zur Bestimmung der funktionalen Größe spezielle Eigenschaften besitzt beziehungsweise sich auf festgelegte Systemarten bezieht, wird FSM im Folgenden nicht an eine konkrete FSM-Methode gebunden. Die folgenden Betrachtungen werden diesbezüglich unter Berücksichtigung der generellen Charakteristika der funktionalen Größenmessung durchgeführt, was eine Anwendung der Ergebnisse auf jede mögliche FSM-Art ermöglicht.

## 5 FSM & SW-Lebenszyklus

Wie im vorhergehenden Abschnitt bereits festgestellt, besitzt die funktionale Größenmessung aufgrund ihrer Eigenschaft, programmiersprachenunabhängig zu sein, die Fähigkeit, im Lebenszyklus eine durchgehende Unterstützung der Softwareentwicklung zu ermöglichen.

In diesem Zusammenhang ist zu beachten, dass der aus der funktionalen Größe ableitbare Informationsgehalt in den verschiedenen Phasen des Lebenszyklus differiert. Um diese Überlegungen weiter ausführen zu können ist es notwendig sich zu überlegen, wann das Wissen über die Size im Entwicklungsprozess am größten ist, zu welchen Zeitpunkten gemessen und wann nur geschätzt werden kann und ob es möglich ist, diese Messungen beziehungsweise Schätzungen miteinander zu verknüpfen. Diese Überlegungen stellen den Kern des nun folgenden Abschnittes dar.

### **5.1 Wann weiß man viel über die Softwaregröße**

Die Diskrepanz zwischen den Schätzungen beziehungsweise Messungen der Softwaregröße im Verlauf der Softwareentwicklung erfordert eine Untersuchung der Ursachen dieser Unterschiede.

Ein in diesem Zusammenhang auffällender Fakt ist, dass die Abschätzung der Softwaregröße für gewöhnlich nicht so exakt ausfällt, wie die direkte Messung. Als Beispiel kann hier die Entfernungsschätzung gegenüber der Messung genommen werden. Egal wie exakt die Schätzung in diesem Fall auch sein mag, die Messung ergibt in der Regel dennoch einen exakteren Wert.

Wünschenswert bei der Bestimmung der Size ist allerdings ein möglichst geringer Unterschied zwischen Abschätzung und Messung der funktionalen Größe von Software. Um diese Konsistenz der funktionalen Größenmessung im gesamten Lebenszyklus gewährleisten zu können, wurde durch [COSMIC2003] ein Model entworfen, welches die funktionalen Nutzeranforderungen im Vorfeld der Implementierung und ebenso danach darstellt:

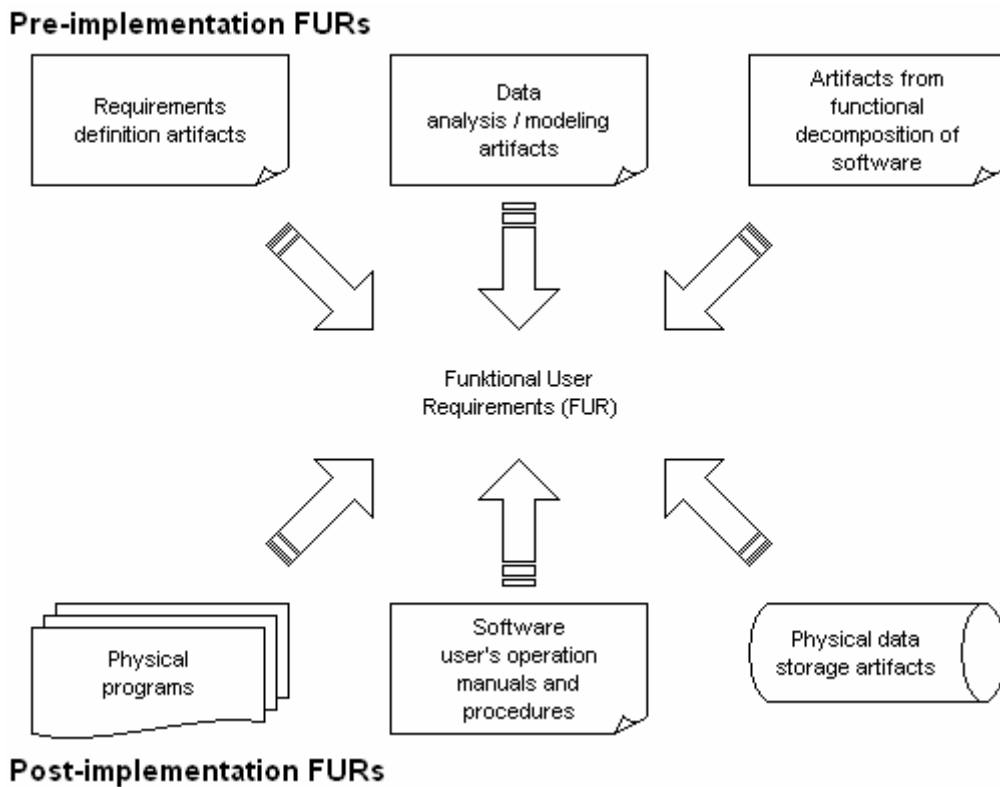


Abbildung 16: Modell für Konsistenzertalt bei funktionaler Größenmessung

Trotz dieser Überlegungen kommt es bei der Entwicklung von Software für gewöhnlich zu einem Anwachsen der funktionalen Größe. Laut [Lothar Diss] sind Ursachen für derartige Abweichungen in sich ändernden Anforderungen, der Zunahme der Funktionalität im Verlauf der Zeit, dem so genannten „function creep“ oder auch in den detaillierteren Informationen, die in späteren Phasen zugänglich sind, zu suchen.

Festzuhalten bleibt aber, dass das Wissen über die Software im Verlauf des Entwicklungsprozesses immer weiter zunimmt (vgl. folgende Analogie):

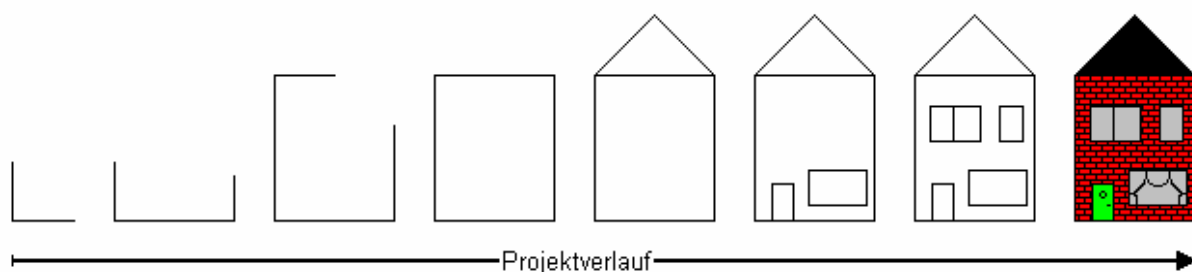


Abbildung 17: Wissensentwicklung mit Projektfortschritt

Dieses Wissen ermöglicht eine größere Exaktheit bei der Abschätzung der Eigenschaften des Endproduktes. So verhält es sich ebenfalls bei der Bestimmung der Softwaregröße. Je weiter

fortgeschritten das Projekt ist, um so detaillierte Informationen sind für die Bestimmung der Size vorhanden. Die meisten Informationen über das Projekt und somit die exaktesten Abschätzungen beziehungsweise Messungen sind demzufolge nach dessen Abschluss vorhanden.

## **5.2 Messung vs. Schätzung**

Entsprechend den Feststellungen des vorhergehenden Abschnittes, nimmt die Exaktheit der Softwaregrößenbestimmung im Verlauf der Softwareentwicklung zu. Wann kann die Größe nun aber nur geschätzt werden und wann kann sie direkt gemessen werden? Die Beantwortung dieser und damit auch der Frage, nach einem Modell, welches Schätzungen und Messungen miteinander kalibriert, ist das Anliegen des nun folgenden Abschnittes.

Schon im Vorfeld dieses Abschnittes wurde festgehalten, dass nur Eigenschaften von Dingen gemessen werden können, die bereits existieren. Ist das noch nicht der Fall, können Schätzungen allerdings einen angenäherten Wert liefern. Dieser Umstand verhält sich bei der Entwicklung von Software und der damit verbundenen Bestimmung des funktionalen Umfangs dieser Software analog.

Da zu Beginn des Entwicklungsprozesses das Endprodukt noch nicht vorhanden ist, sind auch nur wenige Informationen darüber direkt messbar. Aus diesem Grund kann die Softwaregröße hier nur abgeschätzt werden. Diese frühen Schätzungen sind meist noch sehr ungenau. Mit fortschreitender Produktion ist auch immer mehr abzusehen, in welcher Richtung sich die Softwaregröße bewegt, was ebenfalls exaktere Abschätzungen zur Folge hat. Erst nach der Fertigstellung des Software-Produkts lässt sich eine endgültige Aussage bezüglich der Softwaregröße treffen. Das ist darauf zurückzuführen, dass im Verlauf der Entwicklung die Eigenschaften des Endproduktes immer stärker herausgearbeitet werden und diese Informationen ein genaueres Bild ergeben.

Die folgende Abbildung nach [Lothar Diss] verdeutlicht diesbezüglich noch einmal, in welchem Zusammenhang Schätzungen und Messungen während der Projektentwicklung stehen:

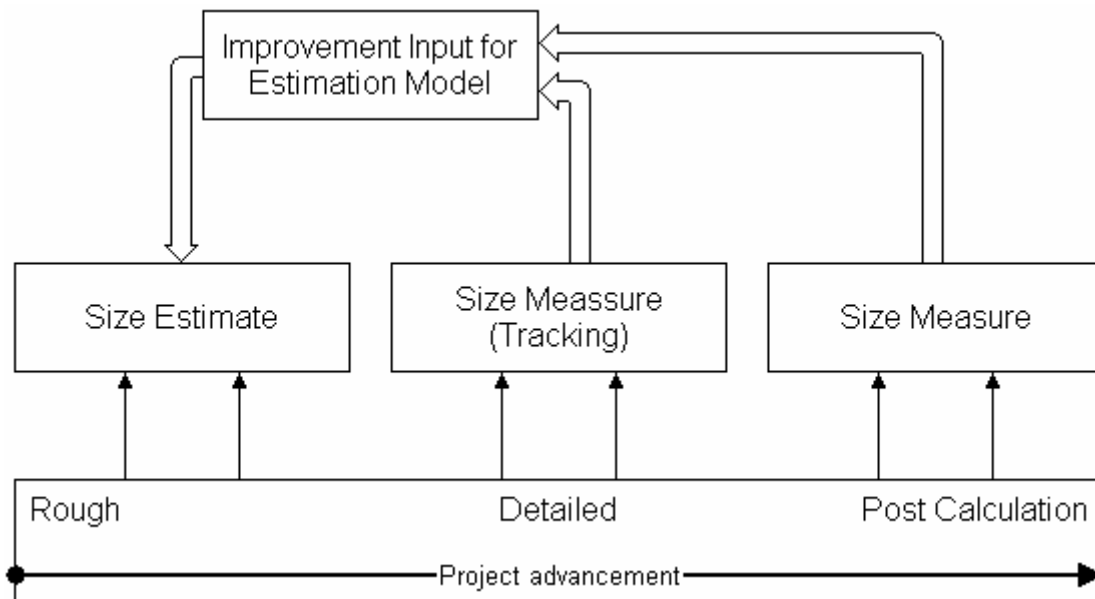


Abbildung 18: Modell zur Kalibrierung von Schätzungen und Messungen

Aus diesem Modell kann ebenso die Information gewonnen werden, in welcher Weise Messungen die Größenabschätzungen unterstützen. Durch die Auswertung der Messergebnisse und der zugehörigen Parameter lässt sich bestimmen, in welcher Weise die Abschätzungen angepasst werden müssen, um bei folgenden Projekten ein der Messung näheres Ergebnis liefern zu können.

In diesem Zusammenhang nimmt die funktionale Größenmessung einen besonderen Stellenwert ein, da sie nicht den entwickelten Code als Berechnungsbasis verwendet, sondern die dem Kunden zur Verfügung gestellten Funktionalitäten. Da das Wissen über diese bereits in frühen Entwicklungsphasen zur Verfügung steht, ist die Messung der funktionalen Größe bereits sehr früh im Entwicklungsprozess sehr genau möglich.

Überlegungen bezüglich der Einbettung dieser Ergebnisse in die bereits im Vorfeld besprochenen Lebenszyklusmodelle erscheinen in diesem Zusammenhang als sinnvoll. Der nun folgende Abschnitt soll sich mit diesen Gedanken näher auseinandersetzen.

### **5.3 Einfluss von Life-Cycle-Modellen**

Im vorhergehenden Abschnitt wurde bereits angedeutet, zu welchen Zeitpunkten innerhalb des Entwicklungszeitraumes gemessen und in welchen nur abgeschätzt werden kann. Dabei ist aufgefallen, dass mit dem Projektfortschritt auch eine Verbesserung der Schätzungen bis hin zur Messung, durch die Zunahme an konkreten Informationen über das System selbst,

hervortritt. Da auch bei Lebenszyklusmodellen eine Zunahme der Informationen zu vermerken ist, erscheint eine Verbindung dieser Ansätze als angebracht. Die Aufgabe dieses Abschnittes wird es nun sein, diese Verbindung zu finden und aufzuzeigen.

Innerhalb der in Kapitel 2.4 aufgezeigten Zyklusphasen gibt es einen unterschiedlichen Bestand an Informationen über das System. Dazu ist allerdings zu sagen, dass im Verlauf der Lebenszyklusphasen die Informationsdichte, und somit wahrscheinlich auch die Korrektheit der funktionalen Größe zunehmen. Bei der Überlegung, in welchen Phasen des Softwarelebenszyklus die funktionale Größe gemessen und wann sie lediglich abgeschätzt werden kann, lassen sich aber zwei verschiedene Herangehensweisen aufzeigen. Diese sind abhängig vom angestrebten Endergebnis der Schätzung beziehungsweise Messung. Gewünschte Ergebnisse können zum einen die Bestimmung der funktionalen Größe selbst, aber auch die näherungsweise oder auch definitive Bestimmung der Eigenschaften des zu entwickelnden Endproduktes sein.

Bei beiden Betrachtungsweisen unterscheiden sich die Überlegungen, in welchen Phasen des Lebenszyklus die funktionale Größe gemessen und wann sie nur abgeschätzt werden kann, was in den nächsten zwei Abschnitten näher untersucht werden soll.

### **5.3.1 Aktuelle funktionale Größe**

Die funktionale Größenmessung besitzt die Eigenschaft, sich außerhalb der programmiertechnischen Gegebenheiten zu bewegen. Aufgrund dieser Basis lässt sich die funktionale Größe einer Software schon sehr früh im Lebenszyklus messen. Die folgende Abbildung soll verdeutlichen, ab welcher Phase des Lebenszyklus sich diese Eigenschaft bemerkbar macht.

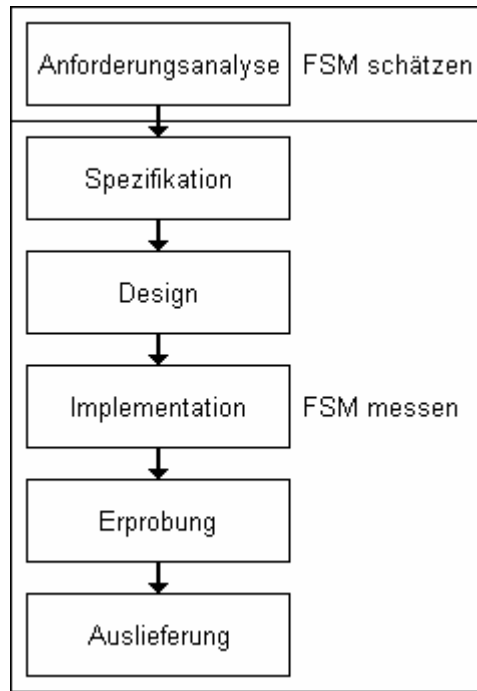


Abbildung 19: Schätzen/Messen der aktuellen funktionalen Größe

Da die funktionale Größe lediglich über die Eigenschaften der in einem Softwareprodukt enthaltenen Funktionalitäten ermitteln lässt, lässt sie sich, wie aus der Abbildung erkennbar ist, ab der Spezifikation messen. Lediglich in der Phase der Anforderungsanalyse, ist eine Abschätzung notwendig. In der Spezifikationsphase sind die Eigenschaften der umzusetzenden Funktionalitäten in den Spezifikationsdokumenten bereits umgesetzt, was eine Messung der funktionalen Größe ermöglicht. Mit dieser Herangehensweise lässt sich allerdings nur der gegenwärtige Zustand des Produktes ermitteln. Rückschlüsse auf die Eigenschaften des Endproduktes lassen sich hieraus allerdings auch nur sehr ungenau bestimmen. Dies bringt uns auch schon zur zweiten Betrachtungsweise der Messung beziehungsweise Abschätzung der funktionalen Größe.

### 5.3.2 Eigenschaften des Endproduktes

Die Eigenschaften des Endproduktes und damit die Bestimmung der Parameter für dessen Entwicklung sind normalerweise die Haupttriebkraft für die Messung von Softwaregrößen. Hierbei ist festzuhalten, dass die Informationsdichte über die Eigenschaften des Endproduktes im Lauf des Lebenszyklus zunimmt. Die folgende Abbildung nach [DumkeSE2] zeigt den Verringerungsprozess der Abweichung von Schätzungen gegenüber der finalen Messung, im Verlauf des Softwarelebenszyklus.

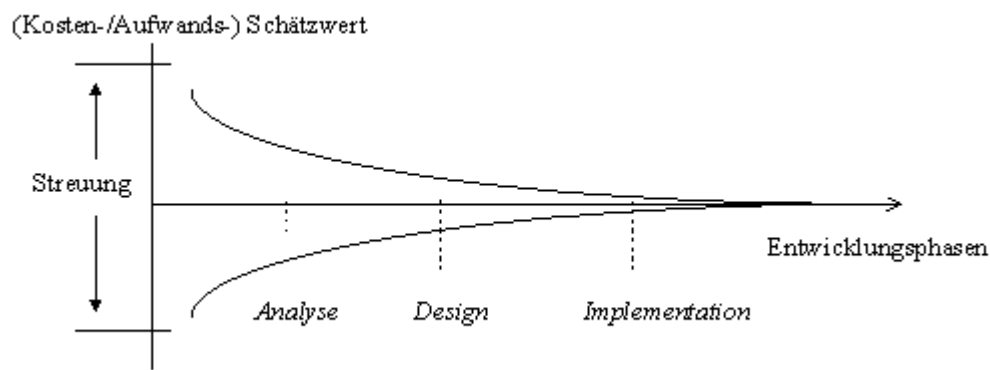


Abbildung 20: Entwicklung von Schätzwerten im Lebenszyklus

Aufgrund dieser Verbesserung des Informationsbestandes und der Eigenschaft nur bereits vorhandenes messen zu können, lässt sich feststellen, dass die direkte Messung der funktionalen Größe des Produktes erst erfolgen kann wenn die Implementierung begonnen hat. Dies beruht auf der Zunahme des function creep während der Softwareentwicklung, nachträglichen Änderungen an den Anforderungen und auch an den detaillierteren Informationen über den exakten Datenaustausch zwischen Nutzer, System und Hardware. Die folgende Abbildung zeigt nun in welchen Phasen des Lebenszyklus eine Schätzung der funktionalen Größe und wann eine Messung erfolgen kann. Hierbei ist jedoch zu beachten, dass das Hauptaugenmerk hierbei auf die Bestimmung der Eigenschaften des Endproduktes gerichtet ist.

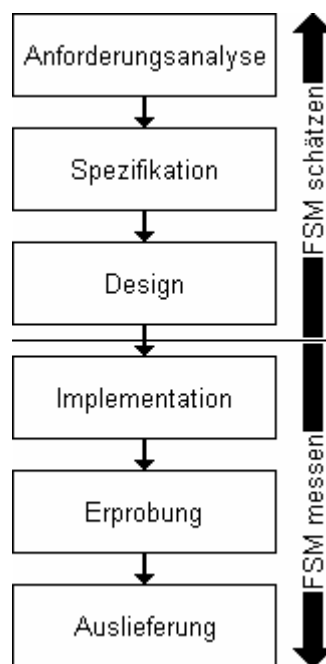


Abbildung 21: Schätzen/Messen der finalen funktionalen Größe

Es ist allerdings aufgrund der vorhergehend besprochenen Eigenschaften der funktionalen Größe festzuhalten, dass die Unterschiede der Schätzungen gegenüber den finalen Messungen sehr gering ausfallen, was letztendlich auch der Hauptgrund für die Nutzung der funktionalen Größe darstellt.

Aufgrund dieser und der vorhergehenden Überlegungen lässt sich nun ein Modell erstellen, welches die Beziehungen zwischen den Phasen des Softwarelebenszyklus mit den Relationen zwischen frühen und späten Phasen der Produktentwicklung zusammenführt. Ein solches Modell könnte folgendermaßen gestaltet sein:

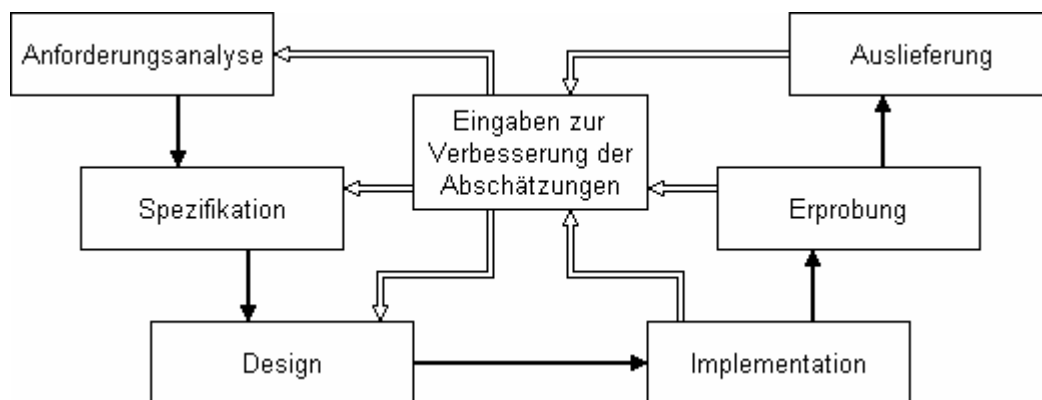


Abbildung 22: Kalibrierung der Messungen im Lebenszyklus

Hier ist eine der in den vorherigen Abschnitten angesprochenen Eigenschaften der funktionalen Größenmessung von größerem Interesse als andere. Es handelt sich hierbei um die gute Kalibrierbarkeit. Die Eingaben, welche aus späten Entwicklungsphasen gegeben werden können verändern hierbei lediglich die Parameter, die Einfluss auf die Bestimmung der funktionalen Größenmessung besitzen. Gemeint ist hiermit sowohl eine Kalibrierung der für den technischen Komplexitätsfaktor notwendigen Faktoren als auch eine angepasste Einstufung der Ein- beziehungsweise Ausgaben entsprechend ihrer Kompliziertheit (vgl. Abbildung 23).

Durch diese Eigenschaften lässt sich eine verbesserte Berechnung der funktionalen Größenmessung bei Durchführung eines ähnlichen Prozesses erzielen.

## 5.4 Zusammenfassung

Der Kerngedanke dieses Abschnittes war eine Überprüfung der Mess- beziehungsweise Abschätzbarkeit der funktionalen Größe im Verlauf der Softwareentwicklung. Die allgemeinen Überlegungen bezüglich der Entwicklung des Informationsgehalts der funktionalen Größe haben diesbezüglich eine Zunahme aufgezeigt. Da Lebenszyklusmodelle

lediglich eine strukturierte Darstellung des Entwicklungsprozesses darstellen, lassen sich diese Erkenntnisse auch auf diese Modelle anwenden.

Weiterhin wurde festgestellt, dass die funktionale Größenmessung bereits in einem sehr frühen Stadium der Entwicklung zur Messung der Softwaregröße eingesetzt werden kann. Lediglich Einflussfaktoren wie beispielsweise function creep oder später eingefügte Anforderungen behindern dabei die exakte Umfangsbestimmung des Endproduktes.

Alle diese Erkenntnisse führen zu dem Schluss, dass mittels der funktionalen Größenmessung ein, über den Lebenszyklus gesehen, ganzheitliches Größenmaß zur Gewinnung der Softwaregröße zur Verfügung steht. Diese vollständige Unterstützung ermöglicht eine Bestimmung des Umfangs des zu entwickelnden Produktes zu jedem Zeitpunkt innerhalb des Entwicklungszeitraumes.

## **6 FSM & CMMI**

### **6.1 Metriken pro CMMI-Stufe**

Da sich die einzelnen Stufen des CMMI in den verschiedenen Schlüsselprozessarealen mit verschiedenen Prozessen, Produkten und Ressourcen befassen, ist es leicht vorstellbar, dass jedem Prozessareal andere Metriken zugeordnet werden können. Die einzige bereits durchgeführte Zuordnung von Metriken zu den Schlüsselprozessarealen des CMMI ist in [Kulpa03] zu finden. Die Untersuchungen dieser Arbeit orientieren sich weitestgehend an diesem im Anhang A beigefügten Metrikenset. Er erhebt keinen Anspruch auf Vollständigkeit oder vollständige Korrektheit, sondern zeigt nur Vorschläge auf. Eine hundertprozentig festgelegte Liste kann nicht existieren, da so gut wie alle Metriken auch in anderen Prozessgebieten zum Einsatz kommen könnten.

In diesem Fall ist die Liste auf die Stufenform des CMMI angepasst, kann aber aufgrund der gleichen Hauptprozessareale auch leicht für die kontinuierliche Form verwendet werden.

Von größerem Interesse ist hier allerdings, ob und in welcher Weise die aufgezeigten Metriken sich mithilfe der funktionalen Größenmessung bestimmen beziehungsweise unterstützen lassen. Der folgende Abschnitt befasst sich mit der Beantwortung dieser Frage. Zu diesem Zweck werden die im Anhang A dargestellten Metriken einzeln entsprechend ihrer Eignung über FSM bestimmt zu werden untersucht.

### **6.2 FSM-basierte Metriken**

Die Aufgabe dieses Kapitels wird es nun sein, herauszufinden, welche der im Anhang A aufgelisteten Metriken auf der funktionalen Größenmessung beruhen und somit unterstützend innerhalb des Softwareentwicklungsprozesses genutzt werden können.

Zu diesem Zweck ist es notwendig sich noch einmal den Zusammenhang zwischen der funktionalen Größenmessung und diesen Metriken zu verinnerlichen. Wie bereits im Abschnitt über funktional size measurement dargestellt, ist die funktionale Größenmessung ein Versuch, die Funktionalität eines Softwareproduktes mit Hilfe der dem Nutzer zur Verfügung gestellten Produkt-Fähigkeiten zu bestimmen. Dies geschieht durch die Bestimmung von „function points“. Unter der Voraussetzung, dass bestimmte Erfahrungswerte oder fortgeschrittenes Wissen über Parameter der zugrundeliegenden Prozessabläufe vorliegen, können, basierend darauf, Abschätzungen von Eigenschaften der

Prozesse bzw. Produkte durchgeführt werden. Auf diese Art und Weise zu bestimmende Software- bzw. Prozesseigenschaften sind unter anderem:

- Kosten
- Aufwand
- Personal Bedarf
- Ressourcen Bedarf
- ...

Ein Beispiel hierfür entnehme ich [Fenton1997]:

Bei einer erkannten Menge von 59 Funktions-Punkten kann, aufgrund des in der Wissensdatenbank gefundenen Aufwands von 2 Mann-Tagen für einen bestimmten Entwickler abgeschätzt werden, dass dieser für die Erstellung des Produktes ungefähr 118 Mann-Tage benötigen wird. Daraus können nun die voraussichtlichen Kosten für personelle Ressourcen für die Entwicklung des Produktes bestimmt werden. Es kann allerdings ebenfalls ermittelt werden, welcher Bedarf an personellen Ressourcen gedeckt werden muss, wenn nur eine begrenzte Entwicklungszeit zur Verfügung steht.

Es kann also der Aufwand bestimmt werden, eine Softwarekomponente zu entwickeln und darauf basierend der Bedarf an Personal und materiellen Ressourcen. Daraus wiederum können dann die Kosten für die jeweilige Entwicklung abgeschätzt bzw. ermittelt werden.

Aufgrund dieser Überlegungen kann nun eine Liste der auf dem functional size measurement beruhenden Metriken erstellt werden. Diese basiert auf der im vorhergehenden Kapitel erstellten Liste der CMMI-Level-zugeordneten Metriken, und ist dementsprechend keinesfalls als vollständig anzusehen.

### **6.3 FSM-basierte Unterstützung für das CMMI**

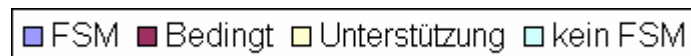
Unter der Bestimmung der funktionalen Größe von Anforderungen ist im Folgenden die Bestimmung der funktionalen Größe der in diesen Anforderungen enthaltenen softwarespezifischen Funktionalitäten zu verstehen. Durch diese Eigenschaft lassen sich Metriken bezüglich der Anforderungen auf Abschätzungen beziehungsweise Messungen der Funktionalen Größe umschlagen!

In [Tabelle 12: werden die in den folgenden Untersuchungen verwendeten Unterscheidungen aufgezeigt. Die jeweiligen Eigenschaften sind dort ebenso ersichtlich.

Unterscheidung	Eigenschaften
nicht durch FSM bestimmbar	Keine Möglichkeit der Bestimmung über FSM
FSM kann unterstützende Metrik sein	Hier kann FSM als unterstützende Metrik, beispielsweise zur Normierung verwendet werden.
bedingt durch FSM bestimmbar	Eine Bestimmung durch FSM kann bei Erfüllung vorhergehender Eigenschaften stattfinden.
teilweise durch FSM bestimmbar	Einige Elemente der Metrik können über FSM bestimmt werden. (wird in Diagrammen zum nächsten Punkt gezählt)
kann durch FSM bestimmt werden.	Eine Bestimmung durch FSM kann stattfinden.

Tabelle 12: Kategorien bei FSM-Unterstützungsuntersuchungen

Die Diagramme unter den Überschriften zeigen die Möglichkeiten der jeweiligen Phase zu einer FSM-Nutzung. Die Darstellung erfolgt in Prozentdarstellung, um eine Normierung zu erhalten. Dabei folgenden Sie diesen Vorgaben:



Die Diagramme sind allerdings nicht als allgemeingültig anzusehen, da sie sich lediglich auf die in den jeweiligen Phasen durch [Kulpa03] vorgeschlagenen Metriken beziehen.

Stattdessen sollten sie als Richtungsweiser für die Nutzung von FSM dienen und andeuten, in welcher Weise eine Unterstützung möglich ist.

### 6.3.1 Level 1

In dieser Stufe existieren keine Hauptprozessareale, innerhalb welcher der Einsatz von Metriken sinnvoll wäre. Aufgrund dessen können hierfür auch keine Metriken festgelegt werden.

### 6.3.2 Level 2

#### 6.3.2.1 Anforderungsmanagement (AM)

Allgemein, kann gesagt werden, dass die Anzahl der Anforderungen weniger Aussagekraft besitzt als die Größe dieser Anforderung. Aus diesem Grund werden in der folgenden Darstellung immer die funktionalen Größen anstelle der Anzahlen verwendet.


<b>Anzahl der Anforderungen</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	Anforderungen schlagen sich direkt in der FS nieder, deshalb FS der Anforderungen
Parameter:	Keine Parameter erforderlich
<b>Anzahl der Anforderungen nach Typ oder Status</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS / Kategorie
Parameter:	Typen / Status der Anforderungen
<b>Änderungshäufigkeit der Anforderungen</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	Änderungshäufigkeit der FS (kontinuierliche Bestimmung vorausgesetzt)
Parameter:	gepflegte Datenbank mit Messungen der FS vorausgesetzt
<b>Anstieg der Änderungen im Bezug zu den festgestellten Anforderungen</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	Scorecard mit Änderungsanzahlen und funktionaler Größe im Verlauf der Zeit
Parameter:	Änderungsanzahlen und funktionale Größe zu verschiedenen Zeitpunkten der Projekterstellung
<b>Anzahl der Änderungsanfragen pro Monat im Verhältnis zur eigentlichen Anzahl von Anforderungen an das Projekt</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	zu verändernde FS pro Monat / totaler FS
Parameter:	Vorausgesetzt wird die Neubestimmung der funktionalen Größe bei Änderungsanfrage und die Speicherung der gewonnenen Daten
<b>Zeitbedarf, Aufwand und Kosten der Implementierung von Änderungsanfragen</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS * Erfahrungswert für Produktivität = Zeitbedarf
Parameter:	Wissensdatenbank mit Produktivität einzelner Programmierer, Kosten pro Manntag, etc.
<b>Menge und Größe von Änderungsanfragen nach Abschluss der Anforderungsphase</b>	
Einordnung:	teilweise durch FSM bestimmbar
Beispiel:	FS der Änderungen nach Abschluss der Anforderungsphase
Parameter:	Anforderungsphase muss abgeschlossen sein; es muss bekannt sein, welche Änderungsanfragen nach der Anforderungsphase eingegangen sind
<b>Kosten für die Implementierung einer Änderungsanfrage</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	Kosten für ( FS mit – FS ohne)
Parameter:	Erfahrungswerte für Kosten für Entwicklung pro Entwickler
<b>Verhältnis von Änderungsanfragen zur Gesamtmenge an Änderungsanfragen während der Projektlaufzeit</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS änderung n / FS änderung ges.
Parameter:	FS der Änderungen muss gespeichert werden um später verwendet werden zu können
<b>Anzahl der akzeptierten aber nicht implementierten Änderungsanfragen</b>	
Einordnung:	FSM kann unterstützende Metrik sein

Beispiel:	FS der akzeptierten, nicht implementierten Änderungen
Parameter:	Keine Parameter erforderlich
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.2.2 Projektplanung (PP)

<b>Verhältnis von fertig gestellten Meilensteinen zu geplanten</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS der fertiggestellten Anforderungen bis zum Meilenstein / FS des gesamten Projekts
Parameter:	Meilensteine müssen festgelegt worden sein. Kontinuierliche Bestimmung der FS wird vorausgesetzt
<b>Beendet Arbeiten, benötigter Aufwand und Geldmittel im Verhältnis zum Plan</b>	
Einordnung:	Teilweise durch FSM bestimmbar
Beispiel:	Aufwand/Geldmittel der bisher umgesetzten FP / geschätzter Aufwand/Geldbedarf nach Plan
Parameter:	vorhergehende Abschätzung des Aufwand und Geldbedarfs muss durchgeführt worden sein; kontinuierliche Bestimmung der FS der bisher umgesetzten Anforderungen
<b>Anzahl der Überarbeitungen der Projektpläne</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Kosten-, Zeit-, und Aufwandsunterschiede je Planüberarbeitung</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS pro Planungsphase vor / nach der Überarbeitung; Kosten, Aufwand und Zeitbedarf für die Unterschiede
Parameter:	Erfahrungswerte über Kosten, Aufwand, Zeitbedarf nötig
<b>Neuplanungs-Aufwand aufgrund von Änderungsanfragen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Zunahmen des Aufwands zum Management des Projekts im Verhältnis zum Plan</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Häufigkeit, Gründe und Ausmaß von Neuplanungen</b>	
Einordnung:	Teilweise durch FSM bestimmbar
Beispiel:	Häufigkeit der Neubestimmung der FS; FS vor Neuplanung – FS nach Neuplanung kann als Ausmaß der Neuplanung interpretiert werden.
Parameter:	kontinuierliche Bestimmung der FS; Aufzeichnungen vorhergehender Neuplanungsdaten
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.2.3 Projektverfolgung und –steuerung (PVS)

<b>Aufwand und Ressourcenverbrauch für Überwachungstätigkeiten</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Änderungstätigkeiten am Projektplan</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der offenen und geschlossenen Korrekturtätigkeiten und Aktionselemente</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Verhältnis von durchgeführten zu geplanten Meilensteinen</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS der in durchgeführten Meilensteinen erstellten Funktionalitäten / gesamte (abgeschätzte) FS aller Meilensteine
Parameter:	Meilensteine müssen festgelegt worden sein; einzuarbeitende Funktionalitäten pro Meilenstein müssen festgelegt worden sein
<b>Anzahl der im festgesetzten Zeitrahmen erfüllten Projekt-Meilensteine</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl und Typ der durchgeführten Reviews</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Ablauf-, Budget- und Größenunterschiede von geplanten und tatsächlich durchgeführten Reviews</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Soll-Ist-Vergleiche aller Planungs- und Verfolgungselemente</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	Soll-Ist-Vergleich der FS ist Bestandteil dieser Untersuchung
Parameter:	stete Aufschlüsselung der Umgesetzten Funktionspunkte und deren funktionaler Größe
<b>Gesamtunterstützung dieses KPA</b>	
	


### 6.3.2.4 Messung und Analyse (MA)

<b>Anzahl der Projekte, die Fortschritts- und Performanzmessungen durchführen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der angesprochenen Messziele</b>	

Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	


### 6.3.2.5 Management von Lieferantenvereinbarungen (ML)

<b>Kosten der COTS (commercial off-the-shelf) – Produkte</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	bei Überlegungen ob COTS vs. Eigenentwicklung, durch Preis / FS der abgedeckten Funktionspunkte
Parameter:	FS der durch COTS abgedeckten Funktionspunkte
<b>Kosten und Aufwand für die Integration der COTS in das Projekt</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	Aufwand bzw. Kosten, bestimmt über die FS der durch COTS abgedeckten Funktionspunkte * Zeitbedarf für die Integration pro Funktionspunkt
Parameter:	Wissen über Produktivität der Mitarbeiter; FS der durch COTS abgedeckten Funktionspunkte
<b>Anzahl der an den Lieferanten-Anforderungen durchgeführten Änderungen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Kosten- und Ablaufunterschiede pro Lieferantenvertrag</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Kosten der Vertrags-Managementaktivitäten verglichen mit der Planung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Tatsächliche Lieferzeitpunkte von vertraglich festgelegten Produkten verglichen mit der Planung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Tatsächliche Lieferzeitpunkte vom Haupt- an Unterauftragnehmer verglichen mit der Planung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der rechtzeitigen Lieferungen vom Lieferanten verglichen mit dem Vertrag</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl und Schwere von nach der Auslieferung gefundenen Fehlern</b>	
Einordnung:	FSM kann unterstützende Metrik sein

Beispiel:	Anzahl der Fehler / FS
Parameter:	Keine Parameter erforderlich
<b>Anzahl der Abweichungen vom Vertrag um Einhaltung der Planung abzusichern</b>	
Einordnung:	Bedingt durch FSM bestimmbar
Beispiel:	sinnvoll wenn FS-relevante Abweichungen vorgenommen wurden (FS der mehr/weniger realisierten Anforderungen)
Parameter:	FS-Relevanz von Fehlern
<b>Anzahl der durchgeführten Qualitäts-Audits verglichen mit der Planung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der vom obersten Management durchgeführten Reviews zur Einhaltung des Budgets und der Abläufe verglichen mit der Planung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der Vertragsverletzungen durch Lieferanten</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	
	

### 6.3.2.6 Qualitätssicherung von Prozessen und Produkten (QS)

<b>Abschluss von Meilensteinen für QS-Aktivitäten verglichen mit der Planung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Abgeschlossene Arbeiten und Aufwand in QS-Aktivitäten verglichen mit dem Plan</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der Produkt-Audits und Aktivitäten-Reviews verglichen mit der Planung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der durchgeführten Prozess-Audits und –Aktivitäten gemessen an den geplanten</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der Defekte pro Release oder Build</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	Fehler je Release oder Build / FS je Release oder Build
Parameter:	Zuordnung der Fehler zu Releases oder Builds
<b>Zeitverbrauch/Aufwand für Nacharbeiten</b>	
Einordnung:	Bedingt durch FSM bestimmbar

Beispiel:	sinnvoll wenn FS-relevante Nacharbeiten durchgeführt werden mussten (FS der Nacharbeiten und daraus bestimmter Aufwand/Kosten)
Parameter:	FS-Relevanz der Nacharbeiten
<b>Zeitverbrauch/Aufwand für QS pro Life-Cycle-Phase</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der Reviews/Audits im Verhältnis zu den gefundenen Fehlern</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Verhältnis der durch interne Reviews gefundenen Fehler zu den vom Endnutzer nach der Lieferung entdeckten</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der pro Life-Cycle-Phase gefundenen Fehler</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	Fehler je Life-Cycle-Phase / FS je Life-Cycle-Phase
Parameter:	Zuordnung der Fehler zu den einzelnen Phasen; FS der in den Phasen umgesetzten Anforderungen
<b>Anzahl der pro Life-Cycle-Phase eingearbeiteten Fehler</b>	
Einordnung:	Bedingt durch FSM bestimmbar
Beispiel:	sinnvoll wenn FS-relevante Fehler eingearbeitet werden (FS der durch Fehler betroffenen Funktionspunkte)
Parameter:	FS-Relevanz der Fehler; FS der zugehörigen Anforderungen
<b>Verhältnis der niedergeschriebenen Unzulänglichkeiten zu den berichtigten</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der an das oberste Management weitergeleiteten Abweichungen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Modul- / Komponentenkomplexität (McCabe, McClure, and Halstead metrics)</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS der jeweiligen Module/Komponenten
Parameter:	pro Modul/Komponente umgesetzte Anforderungen
<b>Gesamtunterstützung dieses KPA</b>	
	

### 6.3.2.7 Konfigurationsmanagement (KM)

<b>Anzahl der pro Zeiteinheit abgearbeiteten Änderungsanfragen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---

<b>Tatsächliche Abschlüsse von Meilensteinen für KM-Aktivitäten verglichen mit der Planung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Bei KM-Aktivitäten abgeschlossene Tätigkeiten bzw. benötigter Aufwand und Geldmittel</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der Änderungen an Konfigurationselementen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der abgewickelten Konfigurations-Audits</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der Fehler mit Status: „noch nicht beseitigt“</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der Fehler mit Status: „Fehler nicht reproduzierbar“</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der Verletzungen von KM-Prozeduren (entdeckt in Audits)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der fehlenden Problemmeldungen gemessen an der Reparaturrate</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Häufigkeit des Überschreibens von Änderungen durch andere Personen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Änderungshäufigkeit von Vorschlägen durch die Entwicklung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Änderungshäufigkeit der Kategorie, des Typs und der Wichtigkeit</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl an Code-Zeilen, die in Bibliotheken unter Konfigurationskontrolle gespeichert wurden</b>	
Einordnung:	Nicht durch FSM bestimmbar

Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.2.8 Unterstützung für Level 2

Die FSM- Unterstützung innerhalb der zweiten CMMI-Ebene gestaltet sich entsprechend der folgenden Darstellung.

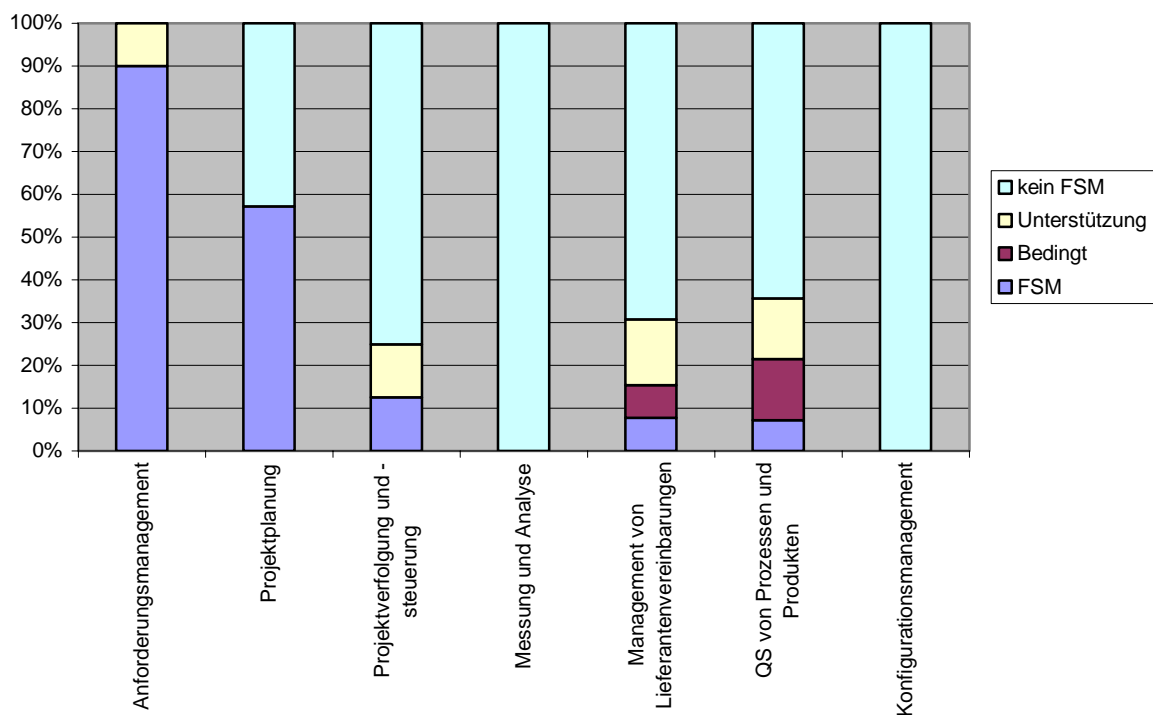


Abbildung 23: FSM-Unterstützung für CMMI-Level 2

Anhand dieser Abbildung ist erkennbar, dass das CMMI-Level 2 in einer Vielzahl der darin enthaltenen Schlüsselprozessareale durch die funktionale Größenmessung unterstützt werden kann. Lediglich im Fall von Messung und Analyse beziehungsweise Konfigurationsmanagement können keine Beziehungen dazu gezeigt werden.

Bezüglich Messung und Analyse muss allerdings erwähnt werden, dass dieser Umstand auf der sehr spezifischen Metrikenabdeckung des durch [Kulpa2001] gegebenen Katalogs beruht. Die Messung der Softwaregröße mittels FSM ist dennoch ein Kerngedanke innerhalb dieses Schlüsselprozessareals. aufgrund dieses Umstands wird die FSM-Unterstützung in diesem Areal in späteren Betrachtungen als gegeben betrachtet.

### 6.3.3 Level 3

#### 6.3.3.1 Anforderungsentwicklung (AE)

<b>Kosten, Ablauf und Aufwand, die für Nacharbeiten verwendet wurden</b>	
Einordnung:	Bedingt durch FSM bestimmbar
Beispiel:	sinnvoll wenn FS-relevante Nacharbeiten durchgeführt werden mussten (FS der Nacharbeiten und daraus bestimmter Aufwand/Kosten)
Parameter:	FS-Relevanz der Nacharbeiten
<b>Fehlerdichte bei Anforderungs-Spezifikationen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der pro Build akzeptierten Anforderungen gemessen an der Gesamtzahl von Anforderungen</b>	
Einordnung:	Teilweise durch FSM bestimmbar
Beispiel:	FS der pro Build akzeptierten softwarespezifischen Anforderungen / FS aller Anforderungen
Parameter:	kontinuierliche Bestimmung der FS von softwarerelevanten Anforderungen; Liste der in den jeweiligen Builds vorhandenen Anforderungen
<b>Tatsächliche Menge an dokumentierten Anforderungen gemessen an der Gesamtzahl abgeschätzter Anforderungen</b>	
Einordnung:	Teilweise durch FSM bestimmbar
Beispiel:	FS der dokumentierten Anforderungen / FS aller Anforderungen
Parameter:	Vorhergehende Abschätzung der FS erforderlich
<b>Gesamt benötigte Arbeitszeit und pro Anforderungsentwicklungs-Aktivität</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anforderungs-Status (Prozentsatz der vorgeschlagenen, überprüften und definierten Spezifikationen)</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	FP pro Spezifikation
Parameter:	Spezifikationen müssen vorliegen
<b>Abschätzung der Anforderungen, des Aufwands für Anforderungsdefinitionen und Anforderungsanalyse bzw. des Ablaufs</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl und Typen von Anforderungsänderungen</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS der Originalanforderung - FS nach Änderung
Parameter:	Bestimmung der FS der einzelnen Anforderungen erforderlich
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.3.2 Technische Umsetzung (TU)

<b>Kosten, Ablauf und Aufwand, die für Nacharbeiten verwendet wurden</b>	
Einordnung:	Bedingt durch FSM bestimmbar
Beispiel:	sinnvoll wenn FS-relevante Nacharbeiten durchgeführt werden mussten (FS der Nacharbeiten und daraus bestimmter Aufwand/Kosten)
Parameter:	FS-Relevanz der Nacharbeiten
<b>Anzahl der im Produkt-/Produktkomponentendesign angesprochenen Anforderungen</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS der in Designphase angesprochenen Anforderungen
Parameter:	Keine Parameter erforderlich
<b>Größe und Komplexität von: Produkt, Produktkomponenten, Schnittstellen und Dokumentationen</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS des Produkts, Komponenten, Schnittstellen (FS von Dokumentationen nicht sinnvoll)
Parameter:	Enthaltene Funktionalitäten müssen bekannt und hinsichtlich Schwierigkeitsgrad gewichtet werden
<b>Fehlerdichte bei Produkten der Technischen Umsetzung (Fehler pro Seite)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der Anforderungen nach Status oder Typ im Verlauf des Projektes</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS der (softwaretechnischen) Anforderungen nach Status oder Typ
Parameter:	Anforderungen mit Status / Typ, Erfahrungswerte bezüglich technischem Komplexitätsfaktor
<b>Problemmeldungen nach Dringlichkeit und Zeitraum bis Beseitigung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der während der Implementierung / Testphase geänderten Anforderungen</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	Änderung der FS durch geänderte Anforderungen (FS <sub>original</sub> - FS <sub>geändert</sub> )
Parameter:	FS der Anforderung vor der Änderung muss bekannt sein → vorhergehende Analysen notwendig
<b>Aufwand für Analyse vorgeschlagener Änderungen pro vorgeschlagene Änderung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der in die Baseline eingearbeiteten Änderungen nach Kategorie</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS der in Baseline eingearbeiteten Änderungen
Parameter:	FS der einzelnen Änderungen muss bekannt sein, Kategorien der Änderungen
<b>Aufwand und Kosten für Implementierung und Test eingearbeiteter Änderungen, incl. Ausgangswerte, Schätzungen und tatsächliche Werte</b>	
Einordnung:	Teilweise durch FSM bestimmbar
Beispiel:	Aufwand/Kosten für Implementierung und Test der Änderungen
Parameter:	Produktivität und Kosten der einzelnen Programmierer und Tester

<b>Schätzungen und tatsächliche Werte der Systemgröße, Wiederverwendung, Aufwand und Ablauf</b>	
Einordnung:	Teilweise durch FSM bestimmbar
Beispiel:	FS des Systems und Aufwand für die Entwicklung können über FSM bestimmt werden
Parameter:	Schätzungen
<b>Abschätzung des gesamten Bedarfs an Mitarbeiterarbeitsstunden für die Systementwicklung nach Aufgabenkategorie und Aktivität</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	$FS_{System} * \text{Produktivität der involvierten Mitarbeiter}$
Parameter:	Produktivität einzelner Mitarbeiter; FS des Systems
<b>Geschätzte sowie tatsächliche Start- bzw. Endzeiten für jede Phase des Lebenszyklus</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	Abschätzung über benötigte Mitarbeiterstunden je Phase (siehe vorhergehender Punkt)
Parameter:	Produktivität einzelner Mitarbeiter; FS des Systems
<b>Anzahl der fertig gestellten Diagramme gemessen an der geschätzten Gesamtanzahl von Diagrammen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der vorgeschlagenen Design-Module / Einheiten</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	Bestimmung der FS der vorgeschlagenen Design-Module / Einheiten sinnvoll
Parameter:	Anzahl der pro vorgeschlagenem Design-Modul / Einheit umgesetzten Anforderungen
<b>Anzahl der ausgelieferten Design-Module / Einheiten</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	Bestimmung der FS der ausgelieferten Design-Module / Einheiten sinnvoll
Parameter:	Anzahl der pro ausgeliefertem Design-Modul / Einheit umgesetzten Anforderungen
<b>Schätzungen und Realwerte der gesamten Codezeilen (neu, modifiziert, wieder verwendet)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	Bestimmung über LOC
Parameter:	Evtl. nachträgliche Bestimmung über umgekehrtes Backfiring (ist jedoch umstritten)
<b>Schätzungen und Realwerte der gesamten Design- und Code-Module / -Einheiten</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	Abschätzung und Messung der FS aller Design- und Code-Module / -Einheiten ist sinnvoll; Weitere Abschätzungen durch andere Size-Measurementformen, zum Beispiel LOC, McCabe sind ebenfalls sinnvoll
Parameter:	Pro Modul/Einheit implementierten Funktionalitäten
<b>Schätzungen und Realwerte der bis zum gegenwärtigen Zeitpunkt insgesamt genutzten CPU-Zeit</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---

<b>Anzahl der codierten und getesteten Code-Einheiten gemessen an den geplanten</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	Anzahl der codierten und getesteten Funktionspunkte / geplante
Parameter:	Anzahlen der codierten, getesteten und geplanten Funktionspunkte
<b>Fehler nach Kategorie, Entdeckungsphase, Einarbeitungsphase, Typ und Dringlichkeit</b>	
Einordnung:	FSM kann unterstützende Metrik sein (Normierung)
Beispiel:	Fehler nach Kategorie / FS
Parameter:	Kategorisierung der Fehler
<b>Schätzungen und Gesamtzahlen von Einheiten, Aufwand und Ablauf</b>	
Einordnung:	Teilweise durch FSM bestimmbar
Beispiel:	Schätzungen des Aufwands über Anforderungen, Messung über System
Parameter:	Aufzeichnungen über Anforderungen, beziehungsweise FS der Anforderungen; Parameter für Bestimmung der FS müssen bekannt sein (Erfahrungswerte)
<b>Geplante, durchgeführte, bestandene und nicht bestandene Systemtests</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Mitgeteilte, beseitigte und nicht beseitigte Widersprüche bei den Tests</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Codewachstum über Prozentsatz des geplanten gegenüber dem tatsächlich implementierten Code</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	Bestimmung des „Funktion creep“ (Wachstum der FS im Zeitverlauf)
Parameter:	kontinuierliche Bestimmung der FS während des Projektverlaufs
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.3.3 Produktintegration (PI)

<b>Produktkomponenten-Integrations-Profile (z.B.: geplante und durchgeführte Produktkomponenten-Zusammenführungen und dabei entdeckte Fehler)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Trends bei der Bewertung der Reports über Probleme während der Integrationsbewertung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Zeitspanne der Öffnung dieser Reports</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.3.4 Verifikation (VE)

<b>Verifikationsprofile (Anzahl der geplanten, durchgeführten Verifikationen und der gefundenen Fehler)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der pro Fehlerkategorie entdeckten Fehler</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Trends bei den Verifikations-Problem-Reports</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Status der Verifikations-Problem-Reports (u. a. Zeitspanne der Öffnung)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der durchgeführten verglichen mit den geplanten Peer-Reviews</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtaufwand für Peer-Reviews verglichen mit dem Plan</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der überprüften Arbeitsprodukte</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.3.5 Validation (VA)

<b>Anzahl der abgeschlossenen Validationsaktivitäten (geplante vs. reale)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Trends in dem Validations-Problem-Reports</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---

<b>Alterung der Validations-Problem-Reports (u. a. Zeitspanne der Öffnung)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.3.6 Organisationsweiter Prozessfokus (OPF)

<b>Anzahl der übermittelten, akzeptierten und implementierten Vorschläge für Prozessverbesserungen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>CMMI-Ebene oder Fähigkeitsgrad</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Abgeschlossene Arbeiten und benötigter Aufwand bzw. Geldmittel bei den Aktivitäten der Organisation zur Einschätzung, Entwicklung und Verbesserung der Prozesse verglichen mit den Planungen auf diesen Gebieten</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Ergebnisse jeder Prozess-Bewertung verglichen mit den Ergebnissen und Empfehlungen vorhergehender Bewertungen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.3.7 Organisationsweite Prozessdefinition (OPD)

<b>Prozentsatz der Projekte, die Prozessarchitekturen und –elemente von den organisationsweit festgelegten Standardprozessen nutzen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Fehlerdichte jedes Prozesselementes aus dem Set der Standardprozesse der Organisation</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der vorgesehenen Meilensteine für Prozessentwicklung und –erhaltung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---

Parameter:	---
<b>Kosten für Prozessdefinitionsaktivitäten</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.3.8 Organisationsweites Training (OT)

<b>Anzahl der gelieferten Trainingskurse (i. A. geplante vs. tatsächliche)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Bewertung der Überprüfungen nach den Trainings</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Trainingsprogramm-Qualitäts-Betrachtungen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Tatsächliche Bereitschaft bei den Trainingskursen verglichen mit der geplanten Bereitschaft</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Fortschritt bei der Verbesserung der Trainingskurse verglichen mit den Projekt-Trainings-Planen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der im Verlauf der Zeit genehmigten Verzichte auf Schulungen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.3.9 Integriertes Projektmanagement (IP)

<b>Anzahl der Änderungen an den definierten Prozessen des Projektes</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Aufwand für die Erstellung des organisationsweiten Sets von Standardprozessen</b>	

Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.3.10 Risikomanagement (RM)

<b>Anzahl der identifizierten, gemanagten, verfolgten und kontrollierten Risiken</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Änderungsaktivitäten für die Pläne zur Risikominderung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der auftauchenden unerwarteten Risiken</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Flüchtigkeit der Risikokategorisierung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Geschätzter und tatsächlicher Aufwand zur Risikominderung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Geschätzte Risikowirkung gemessen an der tatsächlichen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Aufwand und Zeitverbrauch für Risikomanagementaktivitäten gemessen an der Anzahl der tatsächlich vorhandenen Risiken</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Kosten für das Risikomanagement gemessen an den durch tatsächliche Risiken verursachten Kosten</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Tatsächliche Wirkung der identifizierten Risiken verglichen mit den abgeschätzten</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.3.11 Entscheidungsanalyse und -findung (EAF)

Kosten-Nutzen-Rate für die Verwendung formaler Beurteilungsprozesse	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
Gesamtunterstützung dieses KPA	

### 6.3.3.12 Unterstützung für Level 3

Gemäß den Untersuchungen dieses Abschnittes ergibt sich für die FSM-Unterstützung der CMMI-Ebene 3 das in der folgenden Abbildung dargestellte Ergebnis.

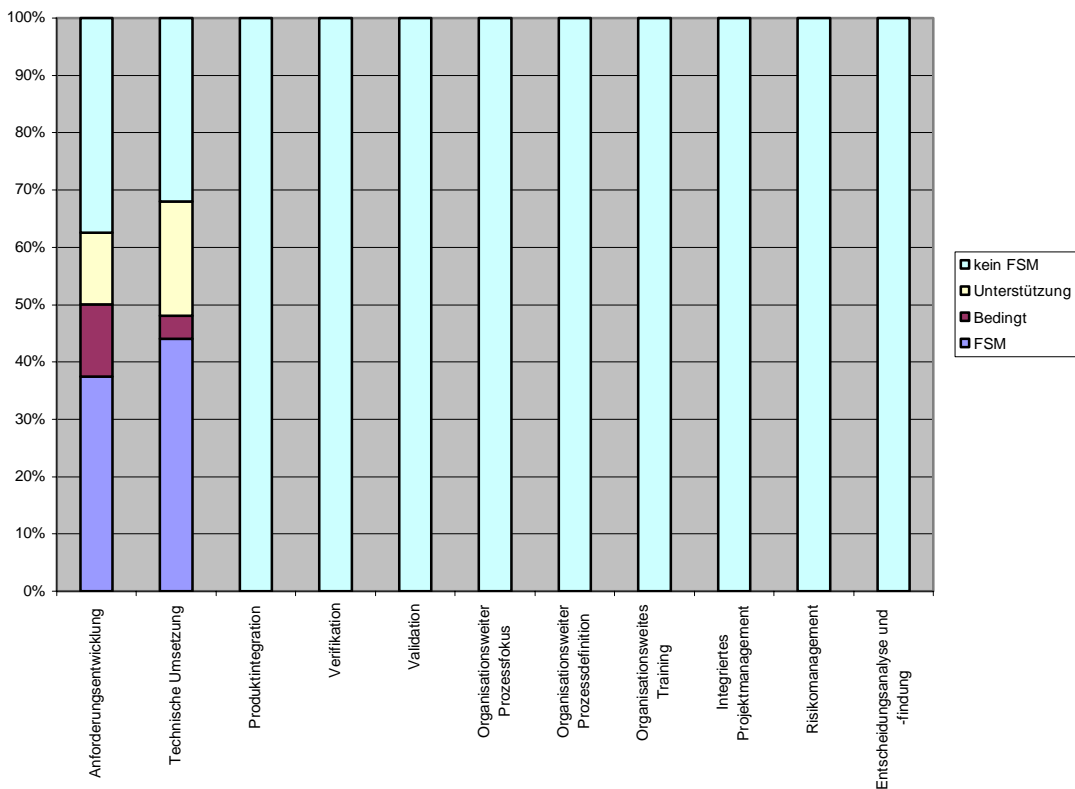


Abbildung 24: FSM-Unterstützung für CMMI-Level 3

Aus dieser Darstellung ist erkennbar, dass die Unterstützung für die dritte CMMI-Ebene sich lediglich auf die Areale Anforderungsentwicklung und Technische Umsetzung bezieht. Hier sollte allerdings bereits festgehalten werden, dass dieser Umstand, ähnlich der nicht vorhandenen FSM-Unterstützung in Level 2 für den Bereich Messung und Analyse, auf der

speziellen Ausprägung der Metriken-Liste nach [Kneuper2001] basiert. Für die Bereiche Verifikation und Validation kann, wie schon im Fall von Messung und Analyse, die Untersuchung der Softwaregröße mittels FSM als Kerngedanke festgelegt werden. Auch bei diesen beiden Arealen wird in späteren Untersuchungen eine FSM-Unterstützung als gegeben angenommen.

## 6.3.4 Level 4

### 6.3.4.1 Performanz der organisationsweiten Prozesse (POP)

<b>Trends in der Prozessperformanz des Unternehmens unter Beachtung von Änderungen an den Arbeitsprodukten und Tätigkeitseigenschaften</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.4.2 Quantitatives Projektmanagement (QPM)

<b>Zeit zwischen Fehlern</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Ausnutzung kritischer Ressourcen</b>	
Einordnung:	Kann durch FSM bestimmt werden
Beispiel:	FS/Hauptspeicher
Parameter:	genutzter Hauptspeicher
<b>Anzahl und Gewicht von Fehlern in veröffentlichten Produkten</b>	
Einordnung:	FSM kann unterstützende Metrik sein (Normierung)
Beispiel:	Anzahl von Fehlern in veröffentlichten Produkten / FS
Parameter:	Anzahl der Fehler in veröffentlichten Produkten
<b>Anzahl und Gewicht von Kundenbeschwerden bezüglich des angebotenen Services</b>	
Einordnung:	Bedingt durch FSM bestimmbar
Beispiel:	sinnvoll wenn wegen Kundenbeschwerden FS-relevante Nacharbeiten durchgeführt werden (FS der Nacharbeiten und daraus bestimmter Aufwand/Kosten)
Parameter:	FS-Relevanz der Nacharbeiten
<b>Anzahl der bei Produktüberprüfungsaktivitäten behobenen Fehler</b>	
Einordnung:	FSM kann unterstützende Metrik sein (Normierung)
Beispiel:	Anzahl der behobenen Fehler / FS
Parameter:	Anzahl bei Produktüberprüfungsaktivitäten behobene Fehler
<b>Rate der übersehenen Fehler</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	Normierung der übersehenen Fehler über pro Funktionspunkt übersehene Fehler

Parameter:	Zuordnung der Fehler zu den Funktionspunkten
<b>Anzahl und Häufigkeit von Fehlern nach Gewicht, während des ersten Jahres nach Auslieferung</b>	
Einordnung:	FSM kann unterstützende Metrik sein (Normierung)
Beispiel:	Anzahl der nach Auslieferung gefundenen Fehler / FS
Parameter:	Anzahl der nach Auslieferung gefundenen Fehler
<b>Zeit für den Zyklus</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Zeitbedarf für Nacharbeiten</b>	
Einordnung:	Bedingt durch FSM bestimmbar
Beispiel:	sinnvoll wenn FS-relevante Nacharbeiten durchgeführt werden mussten (FS der Nacharbeiten und daraus bestimmter Zeitbedarf)
Parameter:	FS-Relevanz der Nacharbeiten
<b>Anforderungsänderungen pro Phase</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	FS der Änderungen pro Phase
Parameter:	Zuordnung der Änderungen zu den Phasen
<b>Raten der abgeschätzten zu den gemessenen Werten von Planungsparametern (z.B. Größe oder Kosten)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Abdeckung und Effizienz von Reviews durch übergestellte Instanzen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Testabdeckung und –effizienz</b>	
Einordnung:	Teilweise durch FSM bestimmbar
Beispiel:	Testabdeckung kann über Anzahl der getesteten Funktionspunkte bestimmt werden
Parameter:	getestete Funktionspunkte müssen bekannt sein
<b>Trainingseffizienz</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Verlässlichkeit (z.B. Durchschnittszeit zwischen Fehlern bei Systemtests)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Prozentsatz der insgesamt eingearbeiteten oder gefundenen Fehler in den unterschiedlichen Phasen des Lebenszyklus</b>	
Einordnung:	FSM kann unterstützende Metrik sein (Normierung)
Beispiel:	Anzahl der Fehler pro Lebenszyklusphase und Kategorie / FS
Parameter:	Anzahl der Fehler nach Lebenszyklusphase und Kategorie
<b>Prozentsatz des Aufwands in den verschiedene Phasen des Projekt-Lebenszyklus</b>	
Einordnung:	Bedingt durch FSM bestimmbar
Beispiel:	Aufwand für Implementierung und Test können über FS der umgesetzten

	Funktionalitäten bezogen auf Produktivität der beteiligten Mitarbeiter bestimmt werden
Parameter:	Produktivität der einzelnen Mitarbeiter/Gruppen
<b>Profile der Unterprozesse unter statistischem Management</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der identifizierten Gründe für Abweichungen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Für Tätigkeiten des quantitativen Prozessmanagements aufgewendete Kosten im Verlauf der Zeit, verglichen mit der Planung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Durchführung von Planungsmeilensteinen für Tätigkeiten des quantitativen Prozessmanagements verglichen mit verglichen mit der genehmigten Planung</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Kosten für schlechte Qualität (z.B. Aufwand für Nacharbeiten)</b>	
Einordnung:	Bedingt durch FSM bestimmbar
Beispiel:	sinnvoll wenn FS-relevante Nacharbeiten durchgeführt werden (FS der Nacharbeiten und daraus bestimmter Aufwand/Kosten)
Parameter:	FS-Relevanz der Nacharbeiten
<b>Kosten für das Erreichen von Qualitätszielen</b>	
Einordnung:	FSM kann unterstützende Metrik sein
Beispiel:	Kosten für die Abschätzungen beziehungsweise Messungen der FS sind Bestandteil dieser Kostenbestimmung
Parameter:	Keine Parameter erforderlich
<b>Gesamtunterstützung dieses KPA</b>	
	

#### 6.3.4.3 Unterstützung für Level 4

Die FSM-Unterstützung gliedert sich für CMMI-Level 4 entsprechend der folgenden, auf den vorhergehenden Überlegungen basierenden, Darstellung auf.

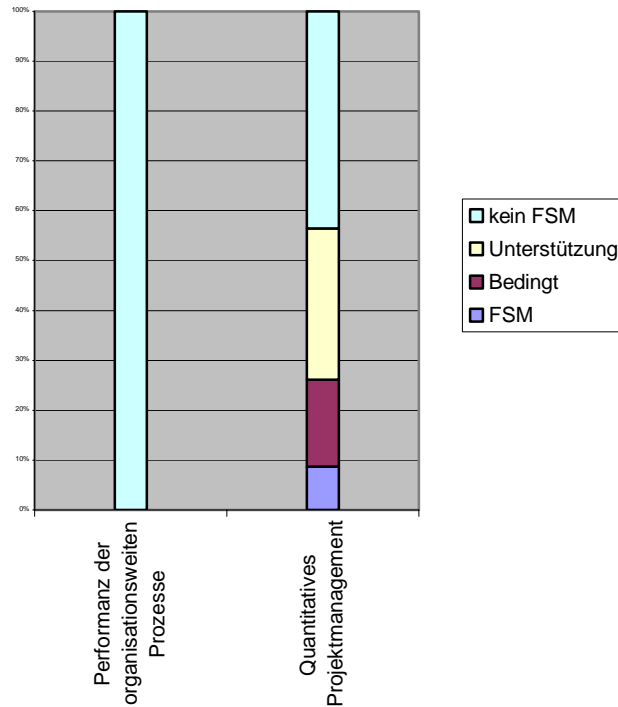


Abbildung 25: FSM-Unterstützung für CMMI-Level 4

Die hier aufgezeigte Unterstützung für das Quantitative Projektmanagement beruht wie zu erkennen ist überwiegend auf der Tatsache, das die Metriken-Liste eine Vielzahl mittels FSM normierbarer Metriken aufweist. Die nichtvorhandene Unterstützung bezüglich der Performanz der organisationsweiten Prozesse, lässt sich durch die Eigenschaft dieses Areals beschreiben, sich nicht mit Produktspezifischen Metriken auseinanderzusetzen. Diese Eigenschaft wird später noch näher erläutert und für weitergehende Betrachtungen herangezogen.

### 6.3.5 Level 5

#### 6.3.5.1 Organisationsweite Innovation und Verbreitung (OIV)

Veränderungen an der Qualität nach den Verbesserungen	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
Veränderungen an der Prozessperformanz nach den Verbesserungen	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
Aktivität zur Änderung der gesamten Technologie (beinhaltet Anzahl, Art und Größe der Änderungen)	
Einordnung:	Nicht durch FSM bestimmbar

Beispiel:	---
Parameter:	---
<b>Effekt der Implementierung von Technologieänderungen verglichen mit den Zielen der Änderungen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der pro Prozessebene übermittelten und implementierten Vorschläge für Prozessverbesserungen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der pro Projekt, Gruppe und Abteilung übermittelten Vorschläge für Prozessverbesserungen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl und Art von Auszeichnungen die jede/s Projekt, Gruppe und Abteilung erhalten hat</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Antwortzeit für die Bearbeitung von Prozessverbesserungsvorschlägen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der pro Berichtszeitraum akzeptierten Prozessverbesserungsvorschläge</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Die gesamte Änderungsaktivität incl. Anzahl, Art und Größe der Änderungen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Effekt der Implementierung jeder Prozessverbesserung verglichen mit ihren definierten Zielen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtperformance der Prozesse der Organisation und Projekte, incl. Effektivität, Qualität und Produktivität verglichen mit den definierten Zielen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtproduktivität und Qualitätstrends für jedes Projekt</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Prozessmessungen die Bezug auf Kundenzufriedenheit nehmen</b>	

Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.5.2 Ursachenanalyse und Problemlösung (UP)

<b>Fehlerdaten (Problemmeldungen, Kundenfehlermeldungen etc.)</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der entfernten Hauptgründe für Fehler</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Änderungen an der Qualität oder Prozessperformanz pro Instanz des Ursachenanalyse- bzw. Lösungsprozesses</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Kosten für Fehlervermeidungsaktivitäten</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Zeit- und Kostenaufwand für die Identifizierung und Korrektur von Fehlern verglichen zu den abgeschätzten Kosten für nicht korrigierte Defekte</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Profilmessungen für vorgeschlagene, geöffnete und fertig gestellte Aktionen</b>	
Einordnung:	Nicht durch FSM bestimmbar
Beispiel:	---
Parameter:	---
<b>Anzahl der pro Stufe eingearbeiteten Fehler</b>	
Einordnung:	FSM kann unterstützende Metrik sein (Normierung)
Beispiel:	Anzahl der eingearbeiteten Fehler Stufe / FS
Parameter:	Anzahl der eingearbeiteten Fehler pro Stufe
<b>Fehleranzahl je Produkt</b>	
Einordnung:	FSM kann unterstützende Metrik sein (Normierung)
Beispiel:	Anzahl der gesamten Fehler / FS
Parameter:	Anzahl der Fehler
<b>Gesamtunterstützung dieses KPA</b>	

### 6.3.5.3 Unterstützung für Level 5

In dieser Darstellung wird die FSM-Unterstützung basierend auf den vorhergehenden Untersuchungen dargestellt.

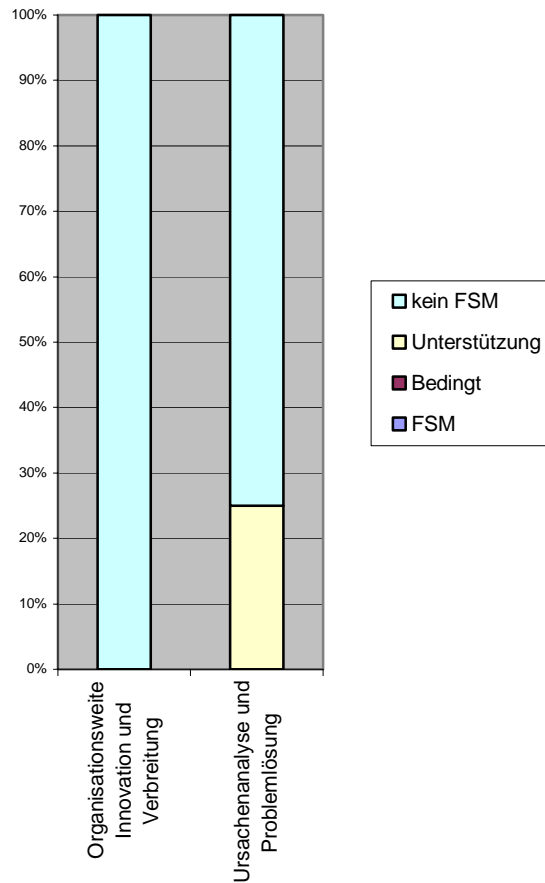


Abbildung 26: FSM-Unterstützung für CMMI-Level 5

Die Eigenschaften der in dieser Darstellung aufgezeigten Areale können entsprechend den Eigenschaften aus der Zusammenfassung der FSM-Unterstützung für CMMI-Level 4 analog übernommen werden. Das bedeutet, dass die Unterstützung, die für Ursachenanalyse und Problemlösung dargestellt wird auf der Normierbarkeit einer Vielzahl der in der Liste enthaltenen Metriken beruht. Die nichtvorhandene Unterstützung für die Organisationsweite Innovation und Verbreitung beruht ebenso auf den nicht-produktspezifischen Metriken dieses Schlüsselprozessareals (vgl. Kapitel 6.3.4.3).

### 6.3.6 Zusammenfassung

Es fällt auf, dass die Vielzahl der durch FSM unterstützten Metriken aus speziellen Bereichen stammt. So ist beispielsweise sehr markant, dass es sich zumeist um Aufwands- bzw. Bedarfsparameter handelt. Eine weitere Gedankenrichtung bei der Auswertung der

funktionalen Größe von Software ist die Bestimmung vorgesehener bzw. durchgeführter Anforderungsänderungen durch das Hinzufügen oder auch Entfernen von Funktionspunkten. Aber auch auf den Funktionalitäten basierende Meilensteine oder entsprechend der Meilensteine bestimmte Funktionspunkte lassen analytische oder statistische Überlegungen und Abschätzungen zu.

Eine Gegenüberstellung der im vorhergehenden Abschnitt festgehaltenen Daten lässt sich anhand folgender Darstellung geben. Zur besseren Analysierbarkeit wurden die Daten durch eine Prozentdarstellung normiert. In den vorhergehenden Kapiteln aufgezeigte Metriken, die teilweise durch funktionale Größenmessung bestimmt werden können, wurden hier auf der FSM-Seite angesiedelt.

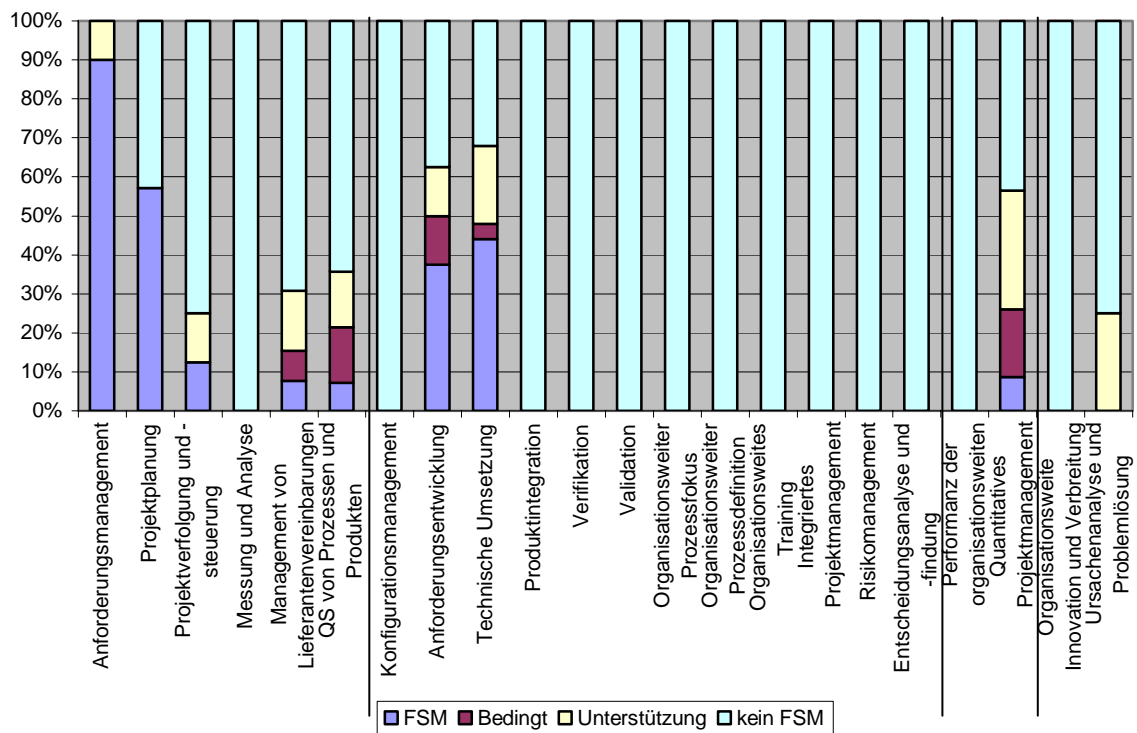


Abbildung 27: FSM-Unterstützung pro KPA

Zu beachten ist jedoch, dass sich die hier dargestellten Diagramme nur auf die Abdeckung des im vorhergehenden Abschnitt aufgezeigten Metrikensets nach [Kulpa03] beziehen. Die nächste Darstellung zeigt die Aufteilung der FSM-Unterstützung in den einzelnen CMMI-Stufen.

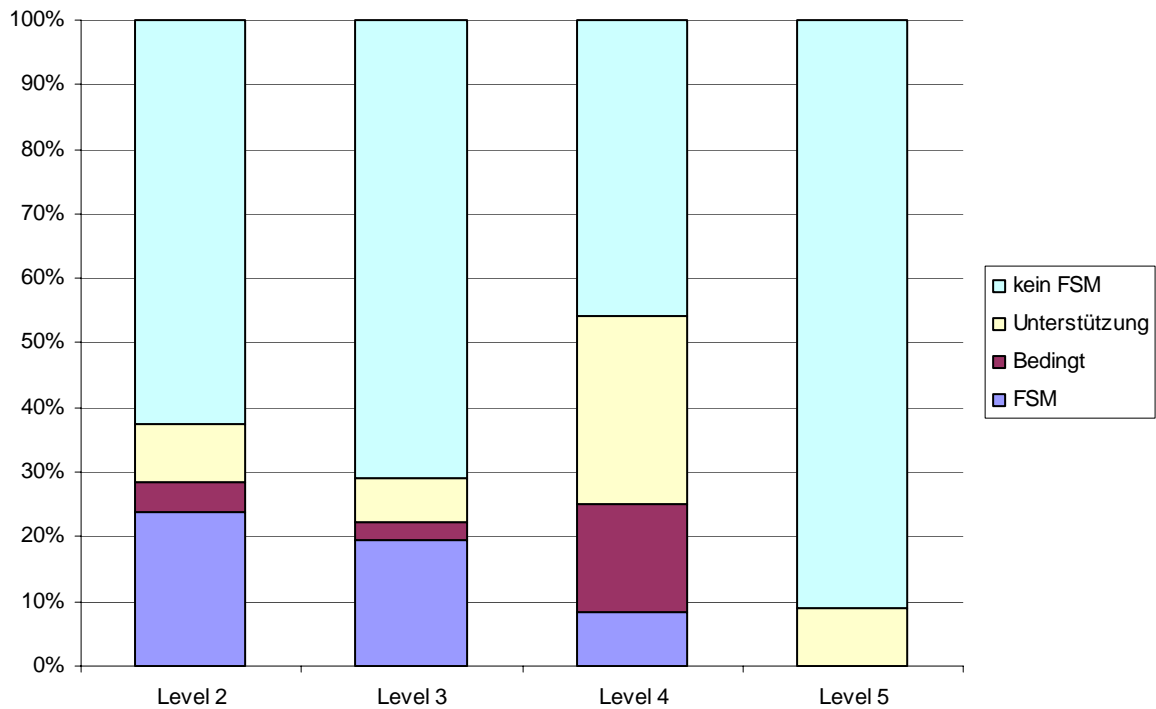


Abbildung 28: Unterstützung der CMMI-Level durch FSM

Hier wird deutlich, dass die Unterstützung, welche die funktionale Größenmessung bieten kann, sich über alle Ebenen des CMMI erstreckt. Weiterhin ist hier erkennbar, welche Ebenen des CMMI-Modells durch die Nutzung der funktionalen Größenmessung in welchem Maß unterstützt werden können. Auffällig ist zudem, dass der Einsatz der FSM mit zunehmender Stufe abnimmt, wohingegen die Menge der durch die Nutzung der FSM unterstützten Metriken zunimmt. Eine Begründung hierfür kann im zunehmenden Prozessbezug der Schlüsselprozessareale mit steigendem CMMI-Level gefunden werden.

Da das Erreichen einer höheren Stufe im CMMI-Modell immer auch das Erfüllen aller Anforderungen der darunter liegenden Ebenen voraussetzt, sind ebenso die Metriken der vorhergehenden Schlüsselprozessareale in den darin vorhanden. Durch die kumulative Graphik in Abbildung 29 wird diese zunehmende Unterstützung durch FSM deutlich. Zu beachten ist hierbei, dass eventuelle Wiederholungen von Metriken in mehreren verschiedenen Schlüsselbereichen keine Berücksichtigung gefunden haben.

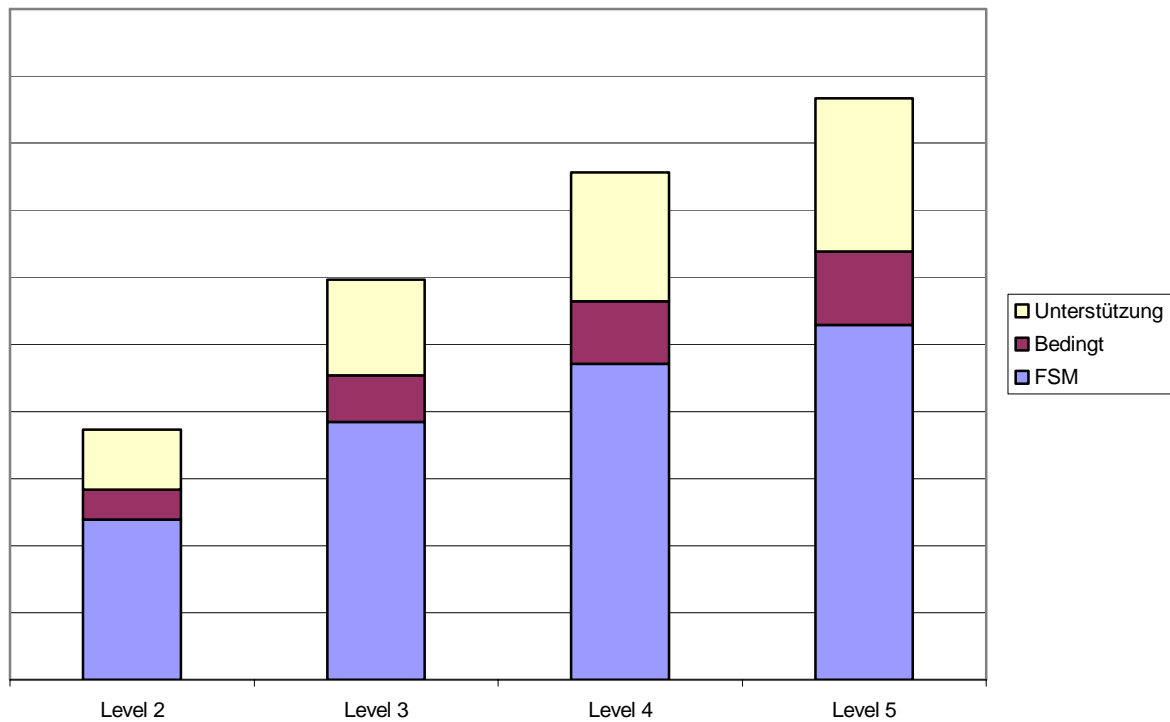


Abbildung 29: Zuwachs der FSM-Unterstützung

Die Verbindung von CMMI und Lebenszyklus-Modellen ist für eine Untersuchung der FSM Unterstützung im CMMI unabdingbar und lässt sich über eine Zuordnung der Schlüsselprozessareale auf die entsprechenden Lebenszykluselemente darstellen. Dieses Mapping soll im nun folgenden Abschnitt näher untersucht werden.

## 7 CMMI & Life-Cycle

Laut [Kneuper2001] schreibt CMMI die Nutzung eines Life-Cycle-Modells vor. Diese Forderung zeigt schon einen Zusammenhang zwischen CMMI und Lebenszyklus auf. Die folgende Abbildung zeigt diese Zuordnung vom Life-Cycle-Modell zu CMMI.

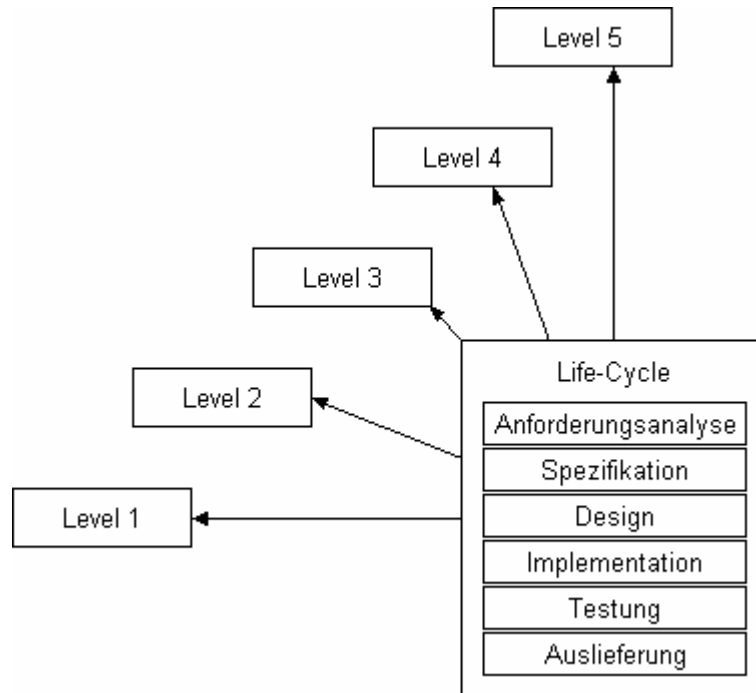


Abbildung 30: Zusammenhang Lifecycle - CMMI

Diese Zuordnung ist sehr trivialer Natur. Die Umkehrung dieser Zusammenführung gestaltet sich hingegen nicht so einfach. Gegenstand dieses Abschnittes soll nun sein, aufzuzeigen, in welcher Art und Weise sich CMMI auf die Lebenszykluselemente mappen lässt.

Zuerst ist festzuhalten, dass sich keine direkte Zuordnung der einzelnen CMMI-Ebenen auf die Elemente des Life-Cycle durchführen lässt. Wäre dies möglich, könnte ein Unternehmen, welches sich in Level 2 befindet weniger Elemente durchlaufen als eines in Level 3.

Eine andere mögliche Herangehensweise wäre eine Zuordnung über die Schlüsselprozessareale der einzelnen Ebenen. Dabei ist jedoch zu beachten, dass sich nicht alle dieser Areale direkt auf Elemente des Lebenszyklus beziehen. Dies liegt an der unterschiedlichen Ausrichtung der einzelnen Ebenen. So sind einige eher projektbezogen, andere aber hingegen unternehmensbezogen. Unter Projektbezug ist hier der Zusammenhang der Areale mit projektspezifischen Erfordernissen zu verstehen. Unternehmensbezug

bezeichnet dabei diejenigen Prozessebenen, die keinen direkten Zusammenhang zur Produktentwicklung aufweisen, sondern Bezug zur Unternehmensführung oder auch zu allgemeinen Prozessen der Produktentwicklung besitzen. Die folgende Abbildung zeigt eine Zuordnung der Areale zu diesen Eigenschaften auf.

<b>CMMI</b>	<b>Projektunterstützung</b>	<b>Unternehmensunterstützung</b>
<b>Level 2</b>	<b>Anforderungsmanagement</b> <b>Projektplanung</b> <b>Projektverfolgung und –steuerung</b> <b>Management von Lieferantenvereinbarungen</b> <b>Messung &amp; Analyse</b> <b>QS von Prozessen und Produkten</b> <b>Konfigurationsmanagement</b>	
<b>Level 3</b>	<b>Anforderungsentwicklung</b> <b>Technische Umsetzung</b> <b>Produktintegration</b> <b>Verifikation</b> <b>Validation</b> <b>Risikomanagement</b> <b>Entscheidungsanalyse und –findung</b>	<b>Organisationsweiter Prozessfokus</b> <b>Organisationsweite Prozessdefinition</b> <b>Organisationsweites Training</b> <b>Integriertes Projektmanagement</b>
<b>Level 4</b>		<b>Performanz organisationsweiter Prozesse</b> <b>Quantitatives Projektmanagement</b>
<b>Level 5</b>		<b>Organisationsweite Innovation und Verbreitung</b> <b>Ursachenanalyse und Problemlösung</b>

Tabelle 13: projektunterstützende vs. unternehmensunterstützende KPAs

Trotz dieser Aufteilung der Schlüsselprozessareale lässt sich eine direkte Zuordnung der Areale mit Produktbezügen auf die Phasen des Lebenszyklus nicht realisieren, da sich viele der Prozessbereiche auf mehrere Phasen beziehen.

Basierend auf den Untersuchungen von [Berauer2003] lassen sich die einzelnen Prozessareale in der folgenden Art und Weise auf die Lebenszyklusphasen übertragen.

	Anforderungsanalyse	Spezifikation	Design	Implementation	Testung	Auslieferung
Anforderungsmanagement						
Projektplanung						
Projektverfolgung und -steuerung						
Management von Lieferantenvereinbarungen						
Messung & Analyse						
QS von Prozessen und Produkten						
Konfigurationsmanagement						
Anforderungsentwicklung						
Technische Umsetzung						
Produktintegration						
Verifikation						
Validation						
Risikomanagement						
Entscheidungsanalyse und -findung						

Abbildung 31: Mapping von KPAs auf Lifecycle-Elemente

Hierbei ist zu beachten, dass die Zusammenstellung generell auf den Phasen des jeweiligen Lebenszyklusmodells beruht und somit je nach Auswahl unterschiedlich ausfallen kann. Um dies zu umgehen, sind hier lediglich die Zusammenhänge zu den bereits in vorhergehenden Abschnitten angesprochenen Hauptelementen jedes Modells berücksichtigt.

Mittels dieser Verbindung zwischen den CMMI-Prozessarealen und den Lifecycle-Elementen lässt sich ein modifiziertes Life-Cycle-Modell erstellen, welches mit folgender Abbildung näher dargestellt wird.

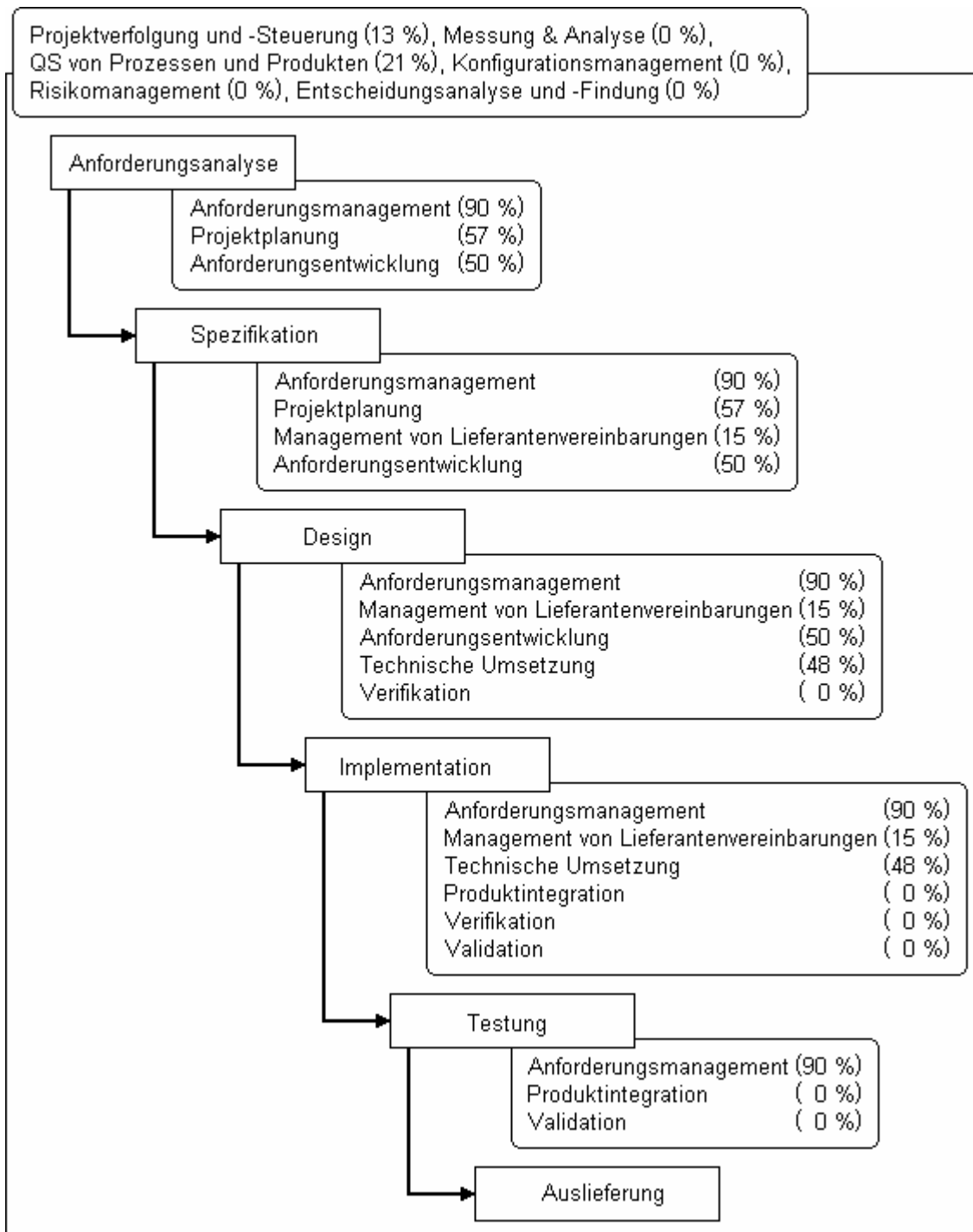


Abbildung 32: Zusammenhang CMMI – Lifecycle

Innerhalb dieses modifizierten Lifecycle-Modells, herrscht aufgrund der im vorhergehenden Abschnitt dargestellten Zusammenhänge eine vollständige Unterstützung durch die funktionale Größenmessung. Die Prozentangaben neben den Schlüsselprozessarealen zeigen nochmals die FSM-Unterstützung bezüglich auf der funktionalen Größenmessung beruhender Metriken innerhalb des Metrikensets nach [Kulpa03] auf. Maße für die FSM eine

unterstützende Metrik sein kann, werden hierbei nicht berücksichtigt, da sie keinen Einfluss auf die vollständige Unterstützung im Lebenszyklus besitzen.

## **7.1 Zusammenfassung**

Das Ziel dieser Arbeit war es, aufzuzeigen, welche Wechselwirkungen zwischen Lebenszyklusmodellen und dem Capability Maturity Model Intergration existieren. Weiterhin sollte die FSM-Unterstützung für Schlüsselprozessareale pro Lebenszyklusphase untersucht werden. Die Betrachtungen haben ergeben, dass eine vollständige Unterstützung des Lebenszyklus durch die funktionale Größenmessung gewährleistet ist. Basierend auf Überlegungen aus Kapitel 6.3.6 wurde gezeigt, dass diese Unterstützung jedoch überwiegend in niedrigen CMMI-Leveln gegeben ist. Dieser Umstand basiert auf der in Tabelle 13 ersichtlichen Zunahme des Prozessbezugs der einzelnen Schlüsselprozesse.

Diese Unterstützung stellt somit einen Grundbaustein der Entwicklung qualitativ hochwertiger Software dar. Die sehr guten Kalibrierbarkeit bei der Berechnung der funktionalen Größe ermöglicht eine weitere Verbesserung bei der Softwareentwicklung. Das in Abbildung 22 dargestellte Modell erlaubt eine kontinuierliche Verbesserung der Abschätzungen beziehungsweise Messungen der funktionalen Größe. Dadurch ermöglicht es beispielsweise auch eine genauere Abschätzung der benötigten Ressourcen oder des Zeitbedarfs.

Mithilfe dieser Erkenntnisse kann basierend auf einer Unternehmensbewertung festgestellt werden, welche Maße zu welchen Zeitpunkten eingesetzt werden können. Basierend darauf, kann ein Modell erstellt werden, wie die für diese Metriken benötigten Parameter ermittelt und verwendet werden können. Ein Beispiel für ein solches Modell soll im nun folgenden Abschnitt am Beispiel der icubic AG eingeführt werden.

## **8 Praktischer Ansatz**

Im Vorfeld der Erstellung eines Messansatzes ist eine Bewertung des betrachteten Unternehmens notwendig, um die zu erreichende CMMI-Ebene bestimmen zu können. Diese Einstufung stellt den Kerngedanken des folgenden Abschnittes dar.

Die eigentliche Herausarbeitung des Messansatzes wird hingegen die Ausrichtung des darauf folgenden Kapitels sein. Dazu werden zuerst die in der herausgearbeiteten CMMI-Stufe vorhandenen KPAs identifiziert.

### **8.1 Vorbetrachtungen und Unternehmensbewertung**

Für den Einsatz der bisher durchgeführten Untersuchungen ist eine vorhergehende Bewertung des betrachteten Unternehmens notwendig. Die Bewertung erfolgt laut [Kneuper2003] am effektivsten auf der Basis eines CMMI-Fragebogens. Dieser Fragebogen ist allerdings nicht allgemein zugänglich, sondern wird lediglich an Pilot-Entwicklungen oder Assessoren ausgegeben.

Aufgrund dessen wird eine Einstufung der icubic AG auf der Basis einer bereits vorhandenen CMM-Bewertung nach [Richter2003] durchgeführt. Diese kann im Anhang B eingesehen werden. Die Übertragbarkeit einer CMM-Bewertung auf eine CMMI-Abschätzung lässt sich auf der Basis von Überlegungen aus [Kneuper2003] bewerkstelligen.

Laut [Kneuper2003] ist der grundlegende Unterschied zwischen den Ebenen des CMMI und CMM-Modells die Einführung eines Prozessgebiets „Messung und Analyse“ im CMMI. Die Inhalte dieser KPA sind im CMM über alle Schlüsselareale verteilt und demzufolge auch in einer CMM-Einschätzung bereits vorhanden.

Ein weiterer Unterschied zwischen beiden Modellen ist die geringere Fokussierung in der CMMI-Stufe 2 auf die Dokumentation der Prozesse. Diese wurde auf Level 3 verschoben, was eine Anwendbarkeit der CMM-Bewertung auf CMMI-relevante Betrachtungen ermöglicht.

Damit sollte gezeigt sein, dass ein Unternehmen, welches sich auf dem Niveau der CMM-Stufe 1 befindet auch bei einer CMMI-Einstufung keine höhere Einstufung erfahren würde.

Wie aus den Betrachtungen in Anhang B erkennbar ist, ist eine Einordnung der icubic AG in Level 2 noch nicht erreicht. Demzufolge ist das betrachtete Unternehmen noch auf Stufe 1 des CMMI-Modells. Aufgrund dieser Basis kann gesagt werden, dass hier lediglich Schritte zum Erreichen der CMMI-Stufe 2 betrachtet werden müssen.

## 8.2 Erstellen einer FSM-basierten Messumgebung für CMMI-Level 2

Bezug nehmend auf Abbildung 32 und mit dem Wissen, dass das Erreichen von CMMI-Ebene 2 das gesetzte Ziel des Unternehmens darstellt, kann ein auf die speziellen Bedürfnisse der icubic AG ausgerichtetes Lebenszyklusmodell entworfen werden. Dieses Modell wird in der folgenden Abbildung aufgezeigt.

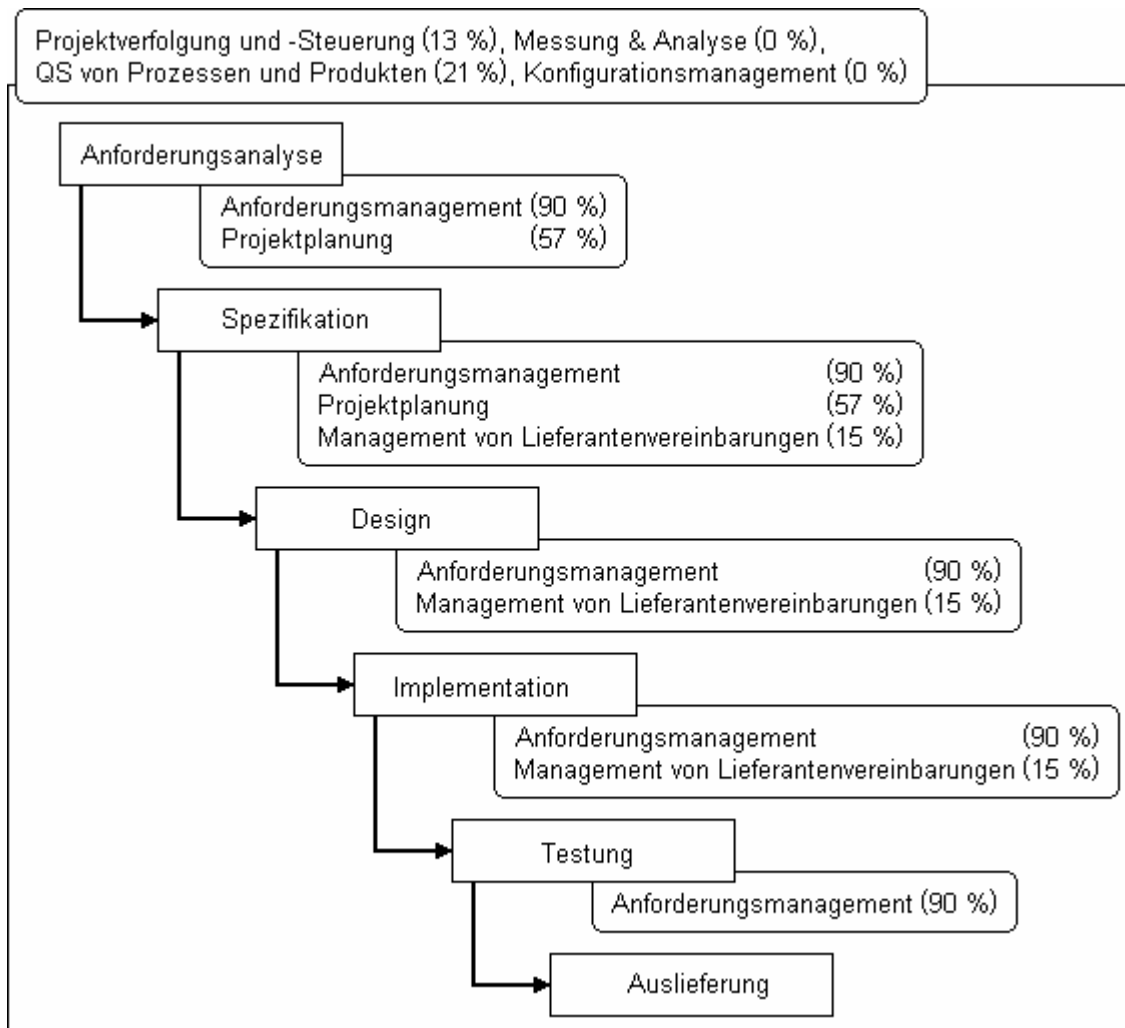


Abbildung 33: FSM-Unterstützung für CMMI-Level 2

Die starke FSM-Unterstützung der zweiten CMMI-Stufe ist in dieser Abbildung ebenso zu erkennen, wie die durchgehende Unterstützung, die durch die funktionale Größenmessung in diesem Level gegeben werden kann.

Aufgrund der Untersuchungen in Kapitel 6.3 können bereits die FSM-unterstützten Metriken herausgearbeitet werden. Die folgende Darstellung zeigt dabei eine auf der Metrikenzusammenstellung von [Fenton1997] beruhende Kollektion dieser Maße.

<b>KPA</b>	<b>FSM-basiert</b>	<b>FSM-normierbar</b>
Anforderungsmanagement	Anzahl der Anforderungen	Anzahl der akzeptierten aber nicht implementierten Änderungsanfragen
	Anzahl der Anforderungen nach Typ oder Status	
	Änderungshäufigkeit der Anforderungen	
	Anstieg der Änderungen im Bezug zu den festgestellten Anforderungen	
	Anzahl der Änderungsanfragen pro Monat im Verhältnis zur eigentlichen Anzahl von Anforderungen an das Projekt	
	Zeitbedarf, Aufwand und Kosten der Implementierung von Änderungsanfragen	
	Menge und Größe von Änderungsanfragen nach Abschluss der Anforderungsphase	
	Kosten für die Implementierung einer Änderungsanfrage	
	Verhältnis von Änderungsanfragen zur Gesamtmenge an Änderungsanfragen während der Projektlaufzeit	
	Projektplanung	
Beendetet Arbeiten, benötigter Aufwand und Geldmittel im Verhältnis zum Plan		
Kosten-, Zeit-, und Aufwandsunterschiede je Planüberarbeitung		
Häufigkeit, Gründe und Ausmaß von Neuplanungen		
Projektverfolgung und -Steuerung	Verhältnis von durchgeführten zu geplanten Meilensteinen FSM-Normalisierbar:	Soll-Ist-Vergleiche aller Planungs- und Verfolgungselemente
Management von Lieferantenvereinbarungen	Kosten und Aufwand für die Integration der COTS in das Projekt	Kosten der COTS (commercial off-the-shelf) – Produkte
	Anzahl der Abweichungen vom Vertrag um Einhaltung der Planung abzusichern	Anzahl und Schwere von nach der Auslieferung gefundenen Fehlern
Qualitätssicherung von Prozessen und Produkten (QS)	Zeitverbrauch/Aufwand für Nacharbeiten	Anzahl der Defekte pro Release oder Build
	Anzahl der pro Life-Cycle-Phase eingearbeiteten Fehler	Anzahl der pro Life-Cycle-Phase gefundenen Fehler
	Modul- / Komponentenkomplexität (McCabe, McClure, and Halstead	

	metrics)	
--	----------	--

Tabelle 14: Metriken für CMMI-Level 2

Aus den Überlegungen in Abschnitt 6.3 können neben den in dieser Tabelle aufgeschlüsselten Messansätzen auch die zugehörigen Parameter bestimmt werden. Um feststellen zu können, zu welchen Zeitpunkten unter den unternehmensspezifischen Gegebenheiten welche dieser Parameter abgegriffen werden können, ist es in erster Linie notwendig, zu ermitteln, welche benötigt werden. Die folgende Tabelle zeigt einen Überblick der zu den aufgezeigten Metriken gehörenden Parameter. Metriken die lediglich mittels FSM normiert werden können werden hier allerdings nicht berücksichtigt, da hierbei die funktionale Größe das Parameter des Maßes darstellt.

<b>Metrik</b>	<b>Parameter</b>
Anzahl (FS) der Anforderungen	Keine Parameter erforderlich
Anzahl (FS) der Anforderungen nach Typ oder Status	Typen / Status der Anforderungen
Änderungshäufigkeit der Anforderungen	gepflegte Datenbank mit Messungen der FS vorausgesetzt
Anstieg der Änderungen im Bezug zu den festgestellten Anforderungen	Änderungsanzahlen und funktionale Größe zu verschiedenen Zeitpunkten der Projekterstellung
Anzahl der Änderungsanfragen pro Monat im Verhältnis zur eigentlichen Anzahl von Anforderungen an das Projekt	Vorausgesetzt wird die Neubestimmung der funktionalen Größe bei Änderungsanfrage und die Speicherung der gewonnenen Daten
Zeitbedarf, Aufwand und Kosten der Implementierung von Änderungsanfragen	Wissensdatenbank mit Produktivität einzelner Programmierer, Kosten pro Manntag, etc.
Menge und Größe von Änderungsanfragen nach Abschluss der Anforderungsphase	Anforderungsphase muss abgeschlossen sein; es muss bekannt sein, welche Änderungsanfragen nach der Anforderungsphase eingegangen sind
Kosten für die Implementierung einer Änderungsanfrage	Erfahrungswerte für Kosten für Entwicklung pro Entwickler
Verhältnis von Änderungsanfragen zur Gesamtmenge an Änderungsanfragen während der Projektlaufzeit	FS der Änderungen muss gespeichert werden um später verwendet werden zu können
Verhältnis von fertig gestellten Meilensteinen zu geplanten	Meilensteine müssen festgelegt worden sein. Kontinuierliche Bestimmung der FS wird vorausgesetzt
Beendetet Arbeiten, benötigter Aufwand und Geldmittel im Verhältnis zum Plan	vorhergehende Abschätzung des Aufwand und Geldbedarfs muss durchgeführt worden sein; kontinuierliche Bestimmung der FS der bisher umgesetzten Anforderungen
Kosten-, Zeit-, und Aufwandsunterschiede je Planüberarbeitung	Erfahrungswerte über Kosten, Aufwand, Zeitbedarf nötig

Häufigkeit, Gründe und Ausmaß von Neuplanungen	kontinuierliche Bestimmung der FS; Aufzeichnungen vorhergehender Neuplanungsdaten
Verhältnis von durchgeführten zu geplanten Meilensteinen	Meilensteine müssen festgelegt worden sein; einzuarbeitende Funktionalitäten pro Meilenstein müssen festgelegt worden sein
Kosten und Aufwand für die Integration der COTS in das Projekt	Wissen über Produktivität der Mitarbeiter; FS der durch COTS abgedeckten Funktionspunkte
Anzahl der Abweichungen vom Vertrag um Einhaltung der Planung abzusichern	FS-Relevanz von Fehlern
Zeitverbrauch/Aufwand für Nacharbeiten	FS-Relevanz der Nacharbeiten
Anzahl der pro Life-Cycle-Phase eingearbeiteten Fehler	FS-Relevanz der Fehler; FS der zugehörigen Anforderungen
Modul- / Komponentenkomplexität (McCabe, McClure, and Halstead metrics)	pro Modul/Komponente umgesetzte Anforderungen

Tabelle 15: FSM-basierte Metriken und Parameter für CMMI-Level 2

Nachdem in Tabelle 15 die benötigten Parameter für die FSM-unterstützten Metriken nach [Kulpa2003] herausgearbeitet wurden, ist es notwendig die Zeitpunkte im Entwicklungsprozess zu identifizieren, an denen die entsprechenden Einflussgrößen bestimmt werden können.

# Anhang A

## Level 2

### Anforderungsmanagement

- Änderungshäufigkeit der Anforderungen
- Anzahl der Anforderungen nach Typ oder Status
- Anstieg der Änderungen im Bezug zu den festgestellten Anforderungen
- Anzahl der Änderungsanfragen pro Monat im Verhältnis zur eigentlichen Anzahl von Anforderungen an das Projekt
- Zeitbedarf, Aufwand und Kosten der Implementierung von Änderungsanfragen
- Menge und Größe von Änderungsanfragen nach Abschluss der Anforderungsphase
- Kosten für die Implementierung einer Änderungsanfrage
- Verhältnis von Änderungsanfragen zur Gesamtmenge an Änderungsanfragen während der Projektlaufzeit
- Anzahl der akzeptierten aber nicht implementierten Änderungsanfragen
- Anzahl der Anforderungen

### Projektplanung

- Verhältnis von fertig gestellten Meilensteinen zu geplanten
- Beendetes Arbeiten, benötigter Aufwand und Geldmittel im Verhältnis zum Plan
- Anzahl der Überarbeitungen der Projektpläne
- Kosten-, Zeit-, und Aufwandsunterschiede je Planüberarbeitung
- Neuplanungs-Aufwand aufgrund von Änderungsanfragen
- Zunahmen des Aufwands zum Management des Projekts im Verhältnis zum Plan
- Häufigkeit, Gründe und Ausmaß von Neuplanungen

### Projektverfolgung und –steuerung

- Aufwand und Ressourcenverbrauch für Überwachungstätigkeiten
- Änderungstätigkeiten am Projektplan
- Anzahl der offenen und geschlossenen Korrekturtätigkeiten und Aktionselemente
- Verhältnis von durchgeführten zu geplanten Meilensteinen
- Anzahl der im festgesetzten Zeitrahmen erfüllten Projekt-Meilensteine
- Anzahl und Typ der durchgeführten Reviews

- Ablauf-, Budget- und Größenunterschiede von geplanten und tatsächlich durchgeführten Reviews
- Soll-Ist-Vergleiche aller Planungs- und Verfolgungselemente

### **Messung und Analyse**

- Anzahl der Projekte, die Fortschritts- und Performanzmessungen durchführen
- Anzahl der angesprochenen Messziele

### **Management von Lieferantenvereinbarungen**

- Kosten der COTS (commercial off-the-shelf) – Produkte
- Kosten und Aufwand für die Integration der COTS in das Projekt
- Anzahl der an den Lieferanten-Anforderungen durchgeführten Änderungen
- Kosten- und Ablaufunterschiede pro Lieferantenvertrag
- Kosten der Vertrags-Managementaktivitäten verglichen mit der Planung
- Tatsächliche Lieferzeitpunkte von vertraglich festgelegten Produkten verglichen mit der Planung
- Tatsächliche Lieferzeitpunkte vom Haupt- an Unterauftragnehmer verglichen mit der Planung
- Anzahl der rechtzeitigen Lieferungen vom Lieferanten verglichen mit dem Vertrag
- Anzahl und Schwere von nach der Auslieferung gefundenen Fehlern
- Anzahl der Abweichungen vom Vertrag um Einhaltung der Planung abzusichern
- Anzahl der durchgeführten Qualitäts-Audits verglichen mit der Planung
- Anzahl der vom obersten Management durchgeführten Reviews zur Einhaltung des Budgets und der Abläufe verglichen mit der Planung
- Anzahl der Vertragsverletzungen durch Lieferanten

### **Qualitätssicherung von Prozessen und Produkten (QS)**

- Abschluss von Meilensteinen für QS-Aktivitäten verglichen mit der Planung
- Abgeschlossene Arbeiten und Aufwand in QS-Aktivitäten verglichen mit dem Plan
- Anzahl der Produkt-Audits und Aktivitäten-Reviews verglichen mit der Planung
- Anzahl der durchgeführten Prozess-Audits und –Aktivitäten gemessen an den geplanten
- Anzahl der Defekte pro Release oder Build
- Zeitverbrauch/Aufwand für Nacharbeiten
- Zeitverbrauch/Aufwand für QS pro Life-Cycle-Phase

- Anzahl der Reviews/Audits im Verhältnis zu den gefundenen Fehlern
- Verhältnis der durch interne Reviews gefundenen Fehler zu den vom Endnutzer nach der Lieferung entdeckten
- Anzahl der pro Life-Cycle-Phase gefundenen Fehler
- Anzahl der pro Life-Cycle-Phase eingearbeiteten Fehler
- Verhältnis der niedergeschriebenen Unzulänglichkeiten zu den berichtigten
- Anzahl der an das oberste Management weitergeleiteten Abweichungen
- Modul- / Komponentenkomplexität (McCabe, McClure, and Halstead metrics)

### **Konfigurationsmanagement (KM)**

- Anzahl der pro Zeiteinheit abgearbeiteten Änderungsanfragen
- Tatsächliche Abschlüsse von Meilensteinen für KM-Aktivitäten verglichen mit der Planung
- Bei KM-Aktivitäten abgeschlossene Tätigkeiten bzw. benötigter Aufwand und Geldmittel
- Anzahl der Änderungen an Konfigurationselementen
- Anzahl der abgewickelten Konfigurations-Audits
- Anzahl der Fehler mit Status: „noch nicht beseitigt“
- Anzahl der Fehler mit Status: „Fehler nicht reproduzierbar“
- Anzahl der Verletzungen von KM-Prozeduren (entdeckt in Audits)
- Anzahl der fehlenden Problemmeldungen gemessen an der Reparaturrate
- Häufigkeit des Überschreibens von Änderungen durch andere Personen
- Änderungshäufigkeit von Vorschlägen durch die Entwicklung
- Änderungshäufigkeit der Kategorie, des Typs und der Wichtigkeit
- Anzahl an Code-Zeilen, die in Bibliotheken unter Konfigurationskontrolle gespeichert wurden

### **Level 3**

#### **Anforderungsentwicklung**

- Kosten, Ablauf und Aufwand, die für Nacharbeiten verwendet wurden
- Fehlerdichte bei Anforderungs-Spezifikationen
- Anzahl der pro Build akzeptierten Anforderungen gemessen an der Gesamtzahl von Anforderungen

- Tatsächliche Menge an dokumentierten Anforderungen gemessen an der Gesamtzahl abgeschätzter Anforderungen
- Gesamt benötigte Arbeitszeit und pro Anforderungsentwicklungs-Aktivität
- Anforderungs-Status (Prozentsatz der vorgeschlagenen, überprüften und definierten Spezifikationen)
- Abschätzung der Anforderungen, des Aufwands für Anforderungsdefinitionen und Anforderungsanalyse bzw. des Ablaufs
- Anzahl und Typen von Anforderungsänderungen

### **Technische Umsetzung**

- Kosten, Ablauf und Aufwand, die für Nacharbeiten verwendet wurden
- Anzahl der im Produkt-/Produktkomponentendesign angesprochenen Anforderungen
- Größe und Komplexität von: Produkt, Produktkomponenten, Schnittstellen und Dokumentationen
- Fehlerdichte bei Produkten der Technischen Umsetzung (Fehler pro Seite)
- Anzahl der Anforderungen nach Status oder Typ im Verlauf des Projektes
- Problemmeldungen nach Dringlichkeit und Zeitraum bis Beseitigung
- Anzahl der während der Implementierung / Testphase geänderten Anforderungen
- Aufwand für Analyse vorgeschlagener Änderungen pro vorgeschlagene Änderung
- Anzahl der in die Baseline eingearbeiteten Änderungen nach Kategorie
- Aufwand und Kosten für Implementierung und Test eingearbeiteter Änderungen, incl. Ausgangswerte, Schätzungen und tatsächliche Werte
- Schätzungen und tatsächliche Werte der Systemgröße, Wiederverwendung, Aufwand und Ablauf
- Abschätzung des gesamten Bedarfs an Mitarbeiterarbeitsstunden für die Systementwicklung nach Aufgabenkategorie und Aktivität
- Geschätzte sowie tatsächliche Start- bzw. Endzeiten für jede Phase des Lebenszyklus
- Anzahl der fertig gestellten Diagramme gemessen an der geschätzten Gesamtanzahl von Diagrammen
- Anzahl der vorgeschlagenen Design-Module / Einheiten
- Anzahl der ausgelieferten Design-Module / Einheiten

- Schätzungen und Realwerte der gesamten Codezeilen (neu, modifiziert, wieder verwendet)
- Schätzungen und Realwerte der gesamten Design- und Code-Module / -Einheiten
- Schätzungen und Realwerte der bis zum gegenwärtigen Zeitpunkt insgesamt genutzten CPU-Zeit
- Anzahl der codierten und getesteten Code-Einheiten gemessen an den geplanten
- Fehler nach Kategorie, Entdeckungsphase, Einarbeitungsphase, Typ und Dringlichkeit
- Schätzungen und Gesamtzahlen von Einheiten, Aufwand und Ablauf
- Geplante, durchgeführte, bestandene und nicht bestandene Systemtests
- Mitgeteilte, beseitigte und nicht beseitigte Widersprüche bei den Tests
- Codewachstum über Prozentsatz des geplanten gegenüber dem tatsächlich implementierten Code

### **Produktintegration**

- Produktkomponenten-Integrations-Profile (z.B.: geplante und durchgeführte Produktkomponenten-Zusammenführungen und dabei entdeckte Fehler)
- Trends bei der Bewertung der Reports über Probleme während der Integrationsbewertung
- Zeitspanne der Öffnung dieser Reports

### **Verifikation**

- Verifikationsprofile (Anzahl der geplanten, durchgeführten Verifikationen und der gefundenen Fehler)
- Anzahl der Fehlerkategorie entdeckten Fehler
- Trends bei den Verifikations-Problem-Reports
- Status der Verifikations-Problem-Reports (u. a. Zeitspanne der Öffnung)
- Anzahl der durchgeführten verglichen mit den geplanten Peer-Reviews
- Gesamtaufwand für Peer-Reviews verglichen mit dem Plan
- Anzahl der überprüften Arbeitsprodukte

### **Validation**

- Anzahl der abgeschlossenen Validationsaktivitäten (geplante vs. reale)
- Trends in dem Validations-Problem-Reports
- Alterung der Validations-Problem-Reports (u. a. Zeitspanne der Öffnung)

### **Organisationsweiter Prozessfokus**

- Anzahl der übermittelten, akzeptierten und implementierten Vorschläge für Prozessverbesserungen
- CMMI-Ebene oder Fähigkeitsgrad
- Abgeschlossene Arbeiten und benötigter Aufwand bzw. Geldmittel bei den Aktivitäten der Organisation zur Einschätzung, Entwicklung und Verbesserung der Prozesse verglichen mit den Planungen auf diesen Gebieten
- Ergebnisse jeder Prozess-Bewertung verglichen mit den Ergebnissen und Empfehlungen vorhergehender Bewertungen

### **Organisationsweite Prozessdefinition**

- Prozentsatz der Projekte, die Prozessarchitekturen und –elemente von den organisationsweit festgelegten Standardprozessen nutzen
- Fehlerdichte jedes Prozesselementes aus dem Set der Standardprozesse der Organisation
- Anzahl der vorgesehenen Meilensteine für Prozessentwicklung und –erhaltung
- Kosten für Prozessdefinitionsaktivitäten

### **Organisationsweites Training**

- Anzahl der gelieferten Trainingskurse (i. A. geplante vs. tatsächliche)
- Bewertung der Überprüfungen nach den Trainings
- Trainingsprogramm-Qualitäts-Betrachtungen
- Tatsächliche Bereitschaft bei den Trainingskursen verglichen mit der geplanten Bereitschaft
- Fortschritt bei der Verbesserung der Trainingskurse verglichen mit den Projekt-Trainings-Planen
- Anzahl der im Verlauf der Zeit genehmigten Verzichte auf Schulungen

### **Integriertes Projektmanagement**

- Anzahl der Änderungen an den definierten Prozessen des Projektes
- Aufwand für die Erstellung des organisationsweiten Sets von Standardprozessen

### **Risikomanagement**

- Anzahl der identifizierten, gemanagten, verfolgten und kontrollierten Risiken
- Änderungsaktivitäten für die Pläne zur Risikominderung
- Anzahl der auftauchenden unerwarteten Risiken

- Flüchtigkeit der Risikokategorisierung
- Geschätzter und tatsächlicher Aufwand zur Risikominderung
- Geschätzte Risikowirkung gemessen an der tatsächlichen
- Aufwand und Zeitverbrauch für Risikomanagementaktivitäten gemessen an der Anzahl der tatsächlich vorhandenen Risiken
- Kosten für das Risikomanagement gemessen an den durch tatsächliche Risiken verursachten Kosten
- Tatsächliche Wirkung der identifizierten Risiken verglichen mit den abgeschätzten

### **Entscheidungsanalyse und –findung**

- Kosten-Nutzen-Rate für die Verwendung formaler Beurteilungsprozesse

### **Level 4**

#### **Performanz der organisationsweiten Prozesse**

- Trends in der Prozessperformanz des Unternehmens unter Beachtung von Änderungen an den Arbeitsprodukten und Tätigkeitseigenschaften

#### **Quantitatives Projektmanagement**

- Zeit zwischen Fehlern
- Ausnutzung kritischer Ressourcen
- Anzahl und Gewicht von Fehlern in veröffentlichten Produkten
- Anzahl und Gewicht von Kundenbeschwerden bezüglich des angebotenen Services
- Anzahl der bei Produktüberprüfungsaktivitäten behobenen Fehler
- Rate der übersehenen Fehler
- Anzahl und Häufigkeit von Fehlern nach Gewicht, während des ersten Jahres nach Auslieferung
- Zeit für den Zyklus
- Zeitbedarf für Nacharbeiten
- Anforderungsänderungen pro Phase
- Raten der abgeschätzten zu den gemessenen Werten von Planungsparametern (z.B. Größe oder Kosten)
- Abdeckung und Effizienz von Reviews durch übergestellte Instanzen
- Testabdeckung und –effizienz
- Trainingseffizienz
- Verlässlichkeit (z.B. Durchschnittszeit zwischen Fehlern bei Systemtests)

- Prozentsatz der insgesamt eingearbeiteten oder gefundenen Fehler in den unterschiedlichen Phasen des Lebenszyklus
- Prozentsatz des Aufwands in den verschiedene Phasen des Projekt-Lebenszyklus
- Profile der Unterprozesse unter statistischem Management
- Anzahl der identifizierten Gründe für Abweichungen
- Für Tätigkeiten des quantitativen Prozessmanagements aufgewendete Kosten im Verlauf der Zeit, verglichen mit der Planung
- Durchführung von Planungsmeilensteinen für Tätigkeiten des quantitativen Prozessmanagements verglichen mit verglichen mit der genehmigten Planung
- Kosten für schlechte Qualität (z.B. Aufwand für Nacharbeiten)
- Kosten für das Erreichen von Qualitätszielen

## **Level 5**

### **Organisationsweite Innovation und Verbreitung**

- Veränderungen an der Qualität nach den Verbesserungen
- Veränderungen an der Prozessperformanz nach den Verbesserungen
- Aktivität zur Änderung der gesamten Technologie (beinhaltet Anzahl, Art und Größe der Änderungen)
- Effekt der Implementierung von Technologieänderungen verglichen mit den Zielen der Änderungen
- Anzahl der pro Prozessebene übermittelten und implementierten Vorschläge für Prozessverbesserungen
- Anzahl der pro Projekt, Gruppe und Abteilung übermittelten Vorschläge für Prozessverbesserungen
- Anzahl und Art von Auszeichnungen die jede/s Projekt, Gruppe und Abteilung erhalten hat
- Antwortzeit für die Bearbeitung von Prozessverbesserungsvorschlägen
- Anzahl der pro Berichtszeitraum akzeptierten Prozessverbesserungsvorschläge
- Die gesamte Änderungsaktivität incl. Anzahl, Art und Größe der Änderungen
- Effekt der Implementierung jeder Prozessverbesserung verglichen mit ihren definierten Zielen
- Gesamtperformanz der Prozesse der Organisation und Projekte, incl. Effektivität, Qualität und Produktivität verglichen mit den definierten Zielen
- Gesamtproduktivität und Qualitätstrends für jedes Projekt

- Prozessmessungen die Bezug auf Kundenzufriedenheit nehmen

### **Ursachenanalyse und Problemlösung**

- Fehlerdaten (Problemmeldungen, Kundenfehlermeldungen etc.)
- Anzahl der entfernten Hauptgründe für Fehler
- Änderungen an der Qualität oder Prozessperformanz pro Instanz des Ursachenanalyse- bzw. Lösungsprozesses
- Kosten für Fehlervermeidungsaktivitäten
- Zeit- und Kostenaufwand für die Identifizierung und Korrektur von Fehlern verglichen zu den abgeschätzten Kosten für nicht korrigierte Defekte
- Profilmessungen für vorgeschlagene, geöffnete und fertig gestellte Aktionen
- Anzahl der pro Stufe eingearbeiteten Fehler
- Fehleranzahl je Produkt

**Anhang B**

# **CMM Bewertung der icubic AG**

von

Sebastian Aisch und Karsten Richter

**Stand: 12.02.2004**

# 1. Einführung

Das Capability Maturity Model für Software, kurz CMM, wurde für den Zweck entwickelt, um die Art und Weise der Softwareentwicklung und –Wartung zu verbessern. Entstanden ist das CMM bei dem Software Engineering Institute (SEI) in Pittsburgh, Pennsylvania, USA mit dem Ziel, die Software-Entwicklungspraktiken in den Vereinigten Staaten zu verbessern. Mittlerweile findet das CMM weltweite Anwendung in den unterschiedlichsten Software-Entwicklungsbereichen und ist zu einem quasi Standard bei der Software-Prozessverbesserung und bei der Software-Prozessqualität geworden.

Für eine junge, aufstrebende Firma, wie die icubic AG, ist es daher unerlässlich, rechtzeitig eine CMM Bewertung durchzuführen, um die Prozessqualität zu ermitteln und dies als Qualitätsmerkmal an die Kunden weiterzugeben. Im Bereich unseres Laborpraktikums wollen wir eine solche CMM Bewertung bei der icubic AG durchführen und aufzeigen, welche Verbesserungen noch notwendig sind, um die nächste CMM Stufe zu erreichen.

## 2. CMM Struktur

### 2.1. Grundlegende Konzepte des CMM

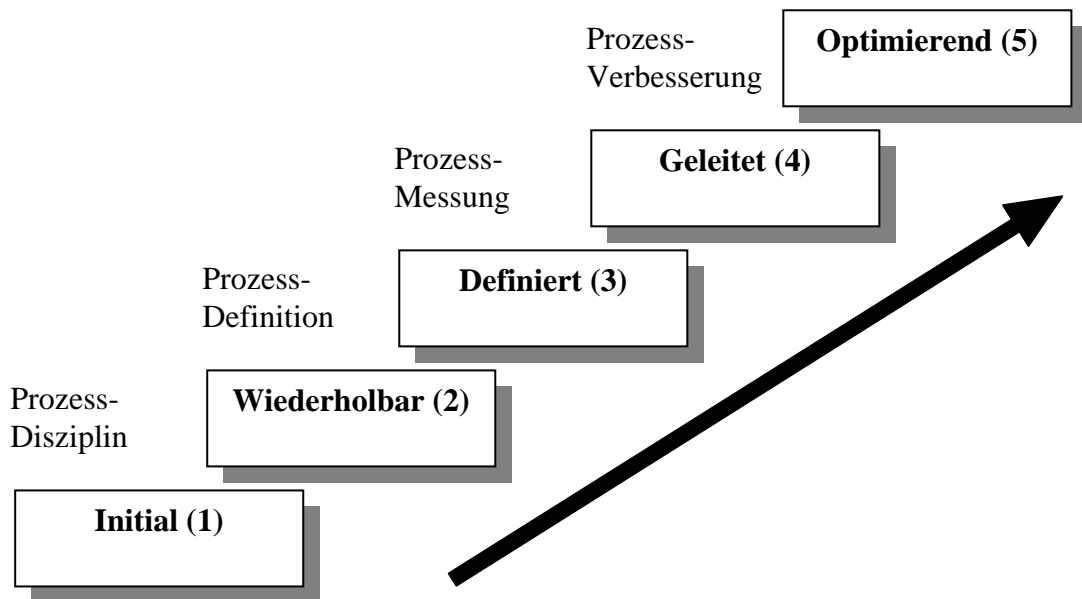
Das Capability Maturity Model ist ein Modell das beschreibt, wie Techniken und Technologien der Softwareentwicklung in Organisationen eingesetzt werden sollen, um die Prozessqualität langfristig zu steigern. Dabei wurde das CMM sehr abstrakt gehalten, um zu gewährleisten, dass es in jedem Software-Unternehmen Anwendung finden kann. Vielmehr bietet es eine fundierte Sammlung von Software-Entwicklungspraktiken, um die Prozesse, die sich mit der Entwicklung und Wartung von Software in einem Unternehmen beschäftigen, ständig zu verbessern. Dabei sind zwei wichtige Bedingungen zu berücksichtigen:

- Alle Arbeitsschritte werden als Prozesse betrachtet und organisiert
- Die Entwicklung des Prozesses wird systematisch geleitet

Mit anderen Worten ist das CMM eine indirekte Möglichkeit zur Bestimmung der Software-Produktqualität. Indirekt deshalb, weil es sich bei diesem Modell nicht um einen Standard zur Bestimmung der Produktqualität an sich handelt, sondern um die Bestimmung der Prozessqualität, mit der man indirekt auf die Produktqualität schließen kann. Im Gegensatz zu dem statischen Standard ISO 9001:2000 beinhaltet das CMM einen Anreiz für Veränderungen.

Die wichtigsten Merkmale des Modells sind die fünf Stufen oder Reifegrade des Software-Entwicklungsprozesses. Ziel dieser Abstufung ist es, mit einer einzelnen Zahl oder einem einzelnen Wort den Reifegrad eines Unternehmens zu beschreiben.

Außerdem wird sehr häufig bei öffentlichen Ausschreibungen von Software-Projekten ein bestimmter Reifegrad des Unternehmens gefordert. Wichtig dabei ist, dass die Schlüsselprozesse der jeweiligen Stufe im Unternehmen umgesetzt wurden. Abstufungen wie 2,98 oder 3,12 machen daher wenig Sinn. Die folgende Abbildung erläutert das Stufenmodell:



Die Stufe 1 ist eigentlich kein richtiger Reifegrad im Sinne des CMMs, da sie keine Anforderungen an die Organisation stellt. Sie stellt lediglich den Ausgangszustand dar. Unternehmen der Stufe 1 besitzen einen ad hoc Software-Entwicklungsprozess, der wenig gelenkt wird und keine definierte Leitung vorgibt. Dies soll nicht bedeuten, dass diese Organisationen nicht in der Lage sind, qualitative Software herzustellen. Allerdings ist das Risiko, das die Kosten, sowohl unter finanziellem als auch personellem Aspekt, zu groß sind, sehr hoch. Demnach sollte jede Organisation, die sich auf Stufe 1 befindet, versuchen, die CMM Stufe 2 zu erreichen, um das Kostenrisiko zu verringern.

Auf Stufe 2, Wiederholbar, können bereits bewältigte Aufgaben mit der gleichen Qualität wiederholt werden. Die wesentliche Voraussetzung dafür ist, dass der Entwicklungsprozess einer Prozessdisziplin untersteht. Damit können Projekte in Bezug auf Kosten, Zeitplan und Anforderungen entsprechend realisiert werden.

Die Stufe 3, Definiert, setzt die Voraussetzungen für erfolgreiche Software-Projekte organisationsweit um. Damit wird der Entwicklungsprozess projektübergreifend als Standard im gesamten Unternehmen definiert und in sich konsistent gehalten.

Auf Stufe 4, Geleitet, wurden die organisationsweiten Prozesse im gesamten Unternehmen implementiert, damit er in seiner Gesamtheit gelenkt und gemessen werden kann.

Auf der Stufe 5, Optimierend, schließlich liegt der Kern bei der ständigen Prozessverbesserung. Alle Software-Prozesse der vorhergehenden Stufen sind Routine und ermöglichen eine Konzentration auf die Prozessverbesserung.

Jede dieser Stufen des CMMs, ausgenommen Stufe 1, stellt bestimmte Anforderungen an die Organisation, die als Schlüsselprozesse bezeichnet werden. Wichtig dabei ist, dass alle Schlüsselprozesse der zu erreichenden Stufe und der darunter liegenden Stufen erfüllt und implementiert sein müssen, um die jeweilige Stufe zu erreichen. Es nützt nichts, alle Anforderungen der Stufe 3 erfüllt zu haben, wenn die Anforderungen der Stufe 2 noch nicht erfüllt sind. Die an die Stufen gebundenen Anforderungen bzw. Schlüsselprozesse sehen wie folgt aus:

Stufe 2 (Wiederholbar):

- Anforderungs-Management (AM)
- Software-Projektplanung (SPP)
- Software-Projektlenkung und –Verfolgung (SPLV)
- Software-Qualitätssicherung (SQS)
- Software-Konfigurations-Management (SKM)
- Software-Unterauftragnehmer-Management (SUM)

Stufe 3 (Definiert):

- Organisationsweiter Prozessfokus (OPF)
- Organisationsweite Prozessdefinition (OPD)
- Trainingsprogramm (TP)
- Integriertes Software-Management (ISM)
- Software-Produkt-Engineering (SPE)
- Gruppen-Koordination (GK)
- Peer-Review (PR)

Stufe 4 (Geleitet):

- Quantitatives Prozess-Management (QPM)
- Software-Qualitäts-Management (SQM)

Stufe 5 (Optimierend):

- Fehlervermeidung (FV)
- Technologie-Change-Management (TCM)
- Prozess-Change-Management (PCM)

Diese Anforderungen bzw. Schlüsselprozesse müssen für das Erreichen der jeweiligen Stufen im Unternehmen implementiert werden. Jeder Schlüsselprozess hat eine bestimmte Anzahl von Schlüsselpraktiken, deren Durchführung angibt, ob die Schlüsselprozesse in der Organisation implementiert wurden oder nicht. Damit ist es möglich, die einzelnen Anforderungen bzw. Schlüsselprozesse und deren Implementation zu überprüfen.

Durch das Erreichen der verschiedenen Stufen, stellt ein Unternehmen sicher, dass ihre Software-Entwicklungsprozesse eine gewisse Qualität aufweisen, welche sich indirekt dann auch auf die Software-Produktqualität auswirkt.

## 2.2. Ziel der CMM Bewertung bei der icubic AG

Was wollen wir mit einer CMM Bewertung bei der icubic AG erreichen? Die icubic AG entwickelt Software für den elektronischen Bondhandel für Großbanken. Dort ist qualitative Software ein Muss. Der Kunde verlangt nicht nur die Funktionstüchtigkeit der Software, sondern auch den Nachweis über die Qualität des Entwicklungsprozesses. Da nach einer ersten Schätzung die icubic AG die zweite Stufe des CMM noch nicht erreicht hat, besteht hier ein großer Interventionsbedarf. Durch das nicht Erreichen der zweiten Stufe zeigt sich eine nicht befriedigende Entwicklungs-Prozessqualität, deren Risiko die icubic AG nicht länger tragen sollte.

Wir wollen mit dieser CMM Bewertung zeigen, auf welcher Stufe die icubic AG steht und die Entwicklungs-Prozesse qualitativ bewerten. Da die icubic AG mit großer Wahrscheinlichkeit den zweiten Reifegrad des CMM nicht erreichen wird, wollen wir zeigen, welche Schlüsselprozesse implementiert, ungenügend implementiert und noch nicht implementiert sind. Damit wird klar, an welchen Stellen der Entwicklungs-Prozess verbessert werden kann und wie diese Verbesserungen umgesetzt werden sollten. Diese Verbesserungen bzw. das Erreichen der nächsten CMM Stufe sollen langfristig den Kostenfaktor, bezogen auf finanzielle und personelle Ressourcen, senken und damit gewährleisten, dass die Software nicht nur qualitativ hervorstechend ist, sondern auch effizienter entwickelt werden kann. Solche Einsparungen lassen sich dann an den Kunden weitergeben, um weiterhin auf dem europäischen Markt konkurrenzfähig zu bleiben.

Nicht zu letzt ist eine solche CMM Bewertung, sofern sie positiv ausfällt, auch ein strategisches Mittel der Kundenakquisition. Je mehr der Kunde über die Qualität eines Entwicklungs-Prozesses weiß, desto größer ist die Wahrscheinlichkeit, dass der Kunde sich für uns entscheidet, da er von der Prozessqualität Rückschlüsse auf die Produktqualität zieht.

## 3. CMM Basisbewertung

Welchen Nutzen bringt uns in diesem Zusammenhang eine CMM Basisbewertung und was beinhaltet sie? Um eine erfolgreiche Verbesserung der Prozessqualität zu bewirken müssen wir uns zuerst einen Überblick über die derzeitige Lage der icubic AG verschaffen. Das versetzt uns dann in die Lage der Firma eine theoretische Position

innerhalb des CMM Stufenmodells zuzuordnen, wodurch wir in den folgenden Abschnitten eine Grundlage für die Verbesserung bzw. Einführung von Prozessen besitzen, die ein Erreichen des nächsten CMM Levels ermöglichen.

Um diese Einstufung vornehmen zu können verwenden wir einen an den CMM Schlüsselprozessen angelehnten Fragenkatalog. Auf die Bedeutung der einzelnen Schlüsselprozesse gehen wir dabei in den einzelnen Abschnitten näher ein.

### 3.1. Anforderungsmanagement (AM)

Durch das Anforderungsmanagement soll die Basis des kompletten weiteren Entwicklungsprozesses gelegt werden. Dabei befasst es sich mit der Erfassung, Verarbeitung und gegebenenfalls Änderung von Kundenanforderungen. Dabei sind alle projektrelevanten Anforderungen zu berücksichtigen. Das bedeutet, dass nicht nur technische Inhalte Beachtung finden, sondern auch in hohem Maße nichttechnische Anforderungen, wie z.B. Lieferzeitpunkte. Diesem Abschnitt fällt eine besondere Bedeutung zu, da er die Basis für Schätzungen, Planungen oder Tätigkeitsverfolgungen im weiteren Softwareentwicklungsprozess darstellt.

*Frage 1:* Bilden die Systemanforderungen mit Software-Bezug die Ausgangsbasis (Baseline) für die Software-Erstellung und das Software-Management?

*Antwort:* Ja! Die Kundenanforderungen werden direkt von den Consultants vor Ort aufgenommen und an das Projektmanagement weitergeleitet. Das Projektmanagement erstellt Zeitpläne zur Einführung der Anforderungen und leitet diese an den Projektleiter weiter.

*Frage 2:* Werden im Falle von Änderungen der Systemanforderungen mit Software-Bezug die Pläne zur Software-Erstellung, die Produkte und die Tätigkeiten angepasst?

*Antwort:* Ja! Hier wird ähnlich der Aufnahme der Systemanforderungen vorgegangen. Änderungen oder Zusätze werden an die Consultants herangetragen und an das Projektmanagement weitergeleitet. Dort werden die Anforderungen in die bestehenden Zeitpläne eingefügt und die so geänderten Systemanforderungen an den Projektleiter weitergeleitet. Der Projektleiter passt daraufhin die Spezifikationen und Arbeitsanweisungen der Programmierer an.

*Frage 3:* Folgt das Projekt einer schriftlichen niedergelegten Vorgabe zum Erstellen und Verwalten der Systemanforderungen mit Software-Bezug?

*Antwort:* Nein! Es existieren keine Vorgaben auf denen das Erstellen und Verwalten von Systemanforderungen basiert.

*Frage 4:* Sind die Personen im Projekt, die mit dem Management der Systemanforderungen betraut sind, auch in dieser Tätigkeit ausgebildet?

*Antwort:* Nein! Die für das Management der Systemanforderungen verantwortlichen Personen sind nicht speziell für diese Tätigkeit geschult worden.

*Frage 5:* Werden Messungen durchgeführt, um den Status der Tätigkeit des Managements der Systemanforderungen festzustellen (z.B. Gesamtzahl der Änderungen der Systemanforderungen, die vorgeschlagen, offen, genehmigt oder in die Baseline implementiert sind)?

*Antwort:* Nein! Der Änderungsstatus wird zwar verwaltet, jedoch nicht für statistische Erhebungen herangezogen.

*Frage 6:* Sind die Tätigkeiten des Managements der Systemanforderungen Reviews durch die Software-Qualitätssicherung unterworfen?

*Antwort:* Nein! Reviews werden generell nicht durch die Software-Qualitätssicherung, sondern sporadisch von der Firmenleitung durchgeführt.

## 3.2. Software-Projektplanung (SPP)

Die Planung der Softwareprojekte hat den Zweck, für Entwicklungstätigkeiten und Projektmanagement relevante Pläne zu erstellen. Das setzt voraus, dass entsprechende Tätigkeiten geplant und ausreichend dokumentiert worden sind. Des Weiteren stellen Abschätzungen von festgelegten Größen (z.B. Zeitbedarf) und deren Dokumentation einen wesentlichen Bestandteil dieses Schlüsselprozesses dar.

*Frage 1:* Werden Schätzungen (z.B. Größe, Kosten, Zeitplan) dokumentiert, die bei der Planung und der Verfolgung des Software-Projektes verwendet werden?

*Antwort:* Nein! Es werden zwar grobe Planungen vorgenommen, aber bei der Planung und Verfolgung des Software-Projektes werden diese kaum dokumentiert oder verwendet.

*Frage 2:* Enthält die dokumentierte Planung die auszuführende Tätigkeit und die eingegangenen Vereinbarungen?

*Antwort:* Ja! Die während des Entwicklungsprozesses eingegangenen Vereinbarungen und notwendigen Tätigkeiten finden sich in der dokumentierten Planung wieder.

*Frage 3:* Halten sich alle beteiligten Gruppen und Einzelpersonen an ihre Vereinbarungen im Rahmen des Software-Projektes?

*Antwort:* Ja! Alle Beteiligten versuchen sich gemäß diesen Vereinbarungen zu verhalten. Sollte die Nichteinhaltung jedoch unumgänglich sein, wird eine entsprechende Anpassung der Projektplanung vorgenommen.

*Frage 4:* Folgt das Projekt einer schriftlich niedergelegten Vorgabe zur Planung eines Software-Projektes?

*Antwort:* Nein! Die Planungstätigkeiten folgen zwar einem wiederkehrenden Schema, basieren jedoch nicht auf einer schriftlich dokumentierten Vorgabe.

*Frage 5:* Werden ausreichende Ressourcen zur Planung des Software-Projektes zur Verfügung gestellt (z.B. Budget, erfahrene Experten)?

*Antwort:* Nein! Es werden zur Planung des Software-Projektes zwar Ressourcen zur Verfügung gestellt, aber nicht in ausreichendem Maße.

*Frage 6:* Werden Messungen zur Bestimmung des Status der Planungstätigkeit des Software-Projektes verwendet (z.B. Meilensteineinhaltung bei Planungstätigkeiten)?

*Antwort:* Nein! Es findet keine Bewertung des Status der Planungstätigkeit auf Basis von Messungen statt. Lediglich bei Auslieferung des Softwareproduktes werden die erreichten Anforderungen den erwarteten gegenübergestellt.

*Frage 7:* Unterzieht der Projektmanager die Planungstätigkeit des Software-Projektes periodischen und ereignisgetriebenen Reviews?

*Antwort:* Ja! Es werden sowohl periodisch als auch durch den Bedarf gesteuert Reviews vom Projektmanagement durchgeführt.

### 3.3. Software-Projektlenkung und –Verfolgung (SPLV)

Durch die Lenkung und Verfolgung des Softwareprojektes soll Kontrolle über den gesamten Entwicklungsprozess erreicht werden. Das wiederum versetzt das Management in die Lage, zu jeder Zeit bei Abweichungen von der eigentlichen Planung in den aktuellen Projektzustand einzugreifen und benötigte Korrekturmaßnahmen durchführen zu können. Möglich wird die Lenkung und Verfolgung des Projektfortschrittes durch Gegenüberstellungen der dokumentierten Schätzungen, Vereinbarungen und Pläne mit den aktuellen Erkenntnissen und durch die darauf basierende Anpassungen der zukünftigen Pläne.

*Frage 1:* Werden die tatsächlichen Ergebnisse und Leistungen (z. B. Größe, Kosten, Zeitplan) mit den Schätzungen der Planung verglichen?

*Antwort:* Nein! Die nur sehr rudimentär vorliegenden Schätzungen der Planungen verhindern einen effektiven Vergleich mit den tatsächlichen Ergebnissen und Leistungen.

*Frage 2:* Werden Korrekturmaßnahmen eingeleitet, wenn die tatsächlichen Ergebnisse und Leistungen signifikant von den Plänen abweichen?

*Antwort:* Ja! Bei auffälligen Abweichungen werden die nicht oder nur ungenügend erreichten Ziele einer erneuten Prüfung unterzogen und wieder in den zukünftigen Entwicklungsablauf eingefügt.

*Frage 3:* Werden Änderungen in den Vereinbarungen von allen beteiligten Gruppen und Einzelpersonen übernommen?

*Antwort:* Ja! Da die Änderungen in den vorhandenen Ablauf eingefügt werden, sind sie dem normalen Ablauf unterworfen. Dies bewirkt eine Übernahme der Änderungen an alle beteiligten Gruppen und Einzelpersonen.

*Frage 4:* Folgt das Projekt einer schriftlich niedergelegten Vorgabe zur Verfolgung und Steuerung seiner Software-Entwicklungstätigkeiten?

*Antwort:* Nein! Es existiert keine schriftliche Vorgabe zur Verfolgung und Steuerung der Software-Entwicklungstätigkeiten.

*Frage 5:* Ist die Verantwortlichkeit im Projekt für die Verfolgung der Fertigstellung von Arbeitsergebnissen und der Tätigkeiten (z. B. Größe, Kosten, Zeitplan) bestimmt?

*Antwort:* Nein! Diese wie auch andere Verantwortlichkeiten innerhalb des Entwicklungsprozesses sind nicht ausreichend bestimmt.

*Frage 6:* Werden Messungen eingesetzt, um den Status der Verfolgungs- und Steuerungstätigkeiten des Software-Projektes festzustellen (z. B. tatsächlich verbrauchter Aufwand bei Verfolgungs- und Steuerungstätigkeiten)?

*Antwort:* Nein! Die Bestimmung des Status der Verfolgungs- und Steuerungstätigkeiten ist keiner Messung unterworfen.

*Frage 7:* Werden die Verfolgungs- und Steuerungstätigkeiten des Software-Projektes periodischen Reviews durch das übergeordnete Management unterzogen (z. B. Projektperformance, Probleme, Risiken, Punkte aus Aktionslisten)?

*Antwort:* Nein! Da nur mangelhafte Verfolgungs- und Steuerungstätigkeiten existieren, werden Reviews nur bedingt durchgeführt.

### 3.4. Software-Qualitätssicherung (SQS)

Durch die Software-Qualitätssicherung soll mit Hilfe von Audits und Reviews eine Transparenz der Prozesse und Produkte des Softwareprojekts für das Management erreicht werden. Zu diesem Zweck müssen die Ergebnisse der Reviews und Audits dem Management in Form von Berichten zur Verfügung gestellt werden. In diesem Zusammenhang soll die Erfüllung der vorgeschriebenen Normen und Verfahren sichergestellt werden.

*Frage 1:* Werden die Tätigkeiten der Software-Qualitätssicherung für das Projekt geplant?

*Antwort:* Ja! Die grundlegenden Tätigkeiten der Software-Qualitätssicherung werden für jedes Software-Projekt geplant und dokumentiert.

*Frage 2:* Liefert die Software-Qualitätssicherung eine objektive Bestätigung, dass die Tätigkeiten und Produkte sich an die vorgeschriebenen Normen, Verfahren und Spezifikationen halten?

*Antwort:* Ja! Die zur Überprüfung der Tätigkeiten und Produkte eingesetzten Dokumente basieren auf den während des Entwicklungsprozesses erstellten Anforderungen und Spezifikationen. Da diese bereits die vorgeschriebenen Normen, Verfahren und Spezifikationen beinhalten, können objektive Bewertungen durchgeführt werden.

*Frage 3:* Werden die Ergebnisse von Reviews und Audits der Software-Qualitätssicherung den beteiligten Gruppen und Einzelpersonen (z. B. denen, die die Arbeit ausführten, und denen, die Verantwortung für die Arbeit tragen) weitergegeben?

*Antwort:* Nein! Da kaum Reviews und Audits von der Qualitätssicherung durchgeführt und dokumentiert werden, können keine Ergebnisse weitergeleitet werden.

*Frage 4:* Werden Abweichungen, die innerhalb des Software-Projektes nicht gelöst wurden, durch das übergeordnete Management behandelt (z. B. Abweichungen von vorgeschriebenen Normen)?

*Antwort:* Ja! Abweichungen, die nicht innerhalb des Software-Projektes gelöst werden können, werden ausschließlich an die übergeordnete Instanz zur Klärung übergeben.

*Frage 5:* Folgt das Projekt einer schriftlich niedergelegten Vorgabe zur Durchführung von Aktivitäten der Software-Qualitätssicherung?

*Antwort:* Nein! Die Software-Qualitätssicherung arbeitet weitestgehend analytisch und außerhalb der Projekte. Schriftlich dokumentierte Vorgaben sind nur ungenügend implementiert.

*Frage 6:* Werden ausreichende Ressourcen zur Durchführung von Software-Qualitätssicherungstätigkeiten zur Verfügung gestellt (z. B. Budget, eine Person, die explizit ernannt wird, um sich mit Abweichungen zu befassen)?

*Antwort:* Ja! Die zur Durchführung von Software-Qualitätssicherungstätigkeiten zur Verfügung gestellten Ressourcen sind vorhanden, verlangen allerdings unter anderem eine variable Personalverteilung.

*Frage 7:* Werden Messungen zur Bestimmung der Kosten und des zeitlichen Status der Software-Qualitätssicherungstätigkeiten durchgeführt (z. B. fertig gestellte Arbeiten, tatsächlich verbrauchter Aufwand und Mittel im Vergleich zum Plan)?

*Antwort:* Nein! Messungen zur Bestimmungen der Kosten und des zeitlichen Status der Software-Qualitätssicherungstätigkeiten werden nur ungenügend durchgeführt, obgleich die Basis für derartige Untersuchungen und Analysen innerhalb der Tätigkeiten und Dokumente bereits gelegt wurde.

*Frage 8:* Werden die Software-Qualitätssicherungstätigkeiten periodischen Reviews durch das übergeordnete Management unterzogen?

*Antwort:* Ja! Die Software-Qualitätssicherungstätigkeiten werden in periodische Reviews durch das übergeordnete Management überprüft.

### 3.5. Software-Konfigurations-Management (SKM)

Der Zweck des Software-Konfigurations-Managements besteht in der Schaffung und Aufrechterhaltung der Integrität von Ergebnissen des Softwareprojekts. Das beinhaltet die Bestimmung der Software-Konfiguration (ausgewählte Software-Arbeitsergebnisse und deren Dokumentation) zu bestimmten festgelegten Zeitpunkten und die Kontrolle bzw. Auswertung der Änderungen an der Konfiguration. Weiterhin muss die Integrität und die Auffindbarkeit der Konfiguration über den gesamten Lebenszyklus hinweg gewahrt werden.

*Frage 1:* Werden die Tätigkeiten des Software-Konfigurationsmanagements für das Projekt geplant?

*Antwort:* Nein, es existiert kein Software-Konfigurationsmanagement.

*Frage 2:* Folgt das Projekt Verfahren des Konfigurationsmanagements bei der Identifizierung der Arbeitsergebnisse, bei geordneten Änderungen daran und beim Bereitstellen der Arbeitsergebnisse?

*Antwort:* Nein, es existiert kein Software-Konfigurationsmanagement.

*Frage 3:* Folgt das Projekt einem schriftlich niedergelegten Verfahren zur Änderungskontrolle bei Konfigurationselementen?

*Antwort:* Nein, es existiert kein Software-Konfigurationsmanagement.

*Frage 4:* Werden Berichte über Software-Baselines (z.B. Protokolle von Sitzungen des Software-Konfigurationskontrollgremiums, zusammenfassende Berichte des Änderungswesens) an die betreffenden Gruppen oder Einzelpersonen verteilt?

*Antwort:* Nein, es existiert kein Software-Konfigurationsmanagement.

*Frage 5:* Folgt das Projekt einer schriftlich niedergelegten Vorgabe zur Durchführung von Tätigkeiten des Software-Konfigurationsmanagements?

*Antwort:* Nein, es existiert kein Software-Konfigurationsmanagement.

*Frage 6:* Sind Mitarbeiter und Manager des Konfigurationsmanagements für ihre Aufgaben im Projekt ausreichend geschult?

*Antwort:* Nein, es existiert kein Software-Konfigurationsmanagement.

*Frage 7:* Werden Messungen zur Bestimmung des Status der Tätigkeiten des Software – Konfigurationsmanagements verwendet (z. B. verbrauchter Aufwand und Budget für Tätigkeiten des Software-Konfigurationsmanagements)?

*Antwort:* Nein, es existiert kein Software-Konfigurationsmanagement.

*Frage 8:* Werden periodisch Audits durchgeführt, um die Übereinstimmung der Software-Baselines mit der zugehörigen Dokumentation nachzuweisen?

*Antwort:* Nein, es existiert kein Software-Konfigurationsmanagement.

### 3.6. Software-Unterauftragsnehmer-Management (SUM)

Das Software-Unterauftragsnehmer-Management soll das Unternehmen in die Lage versetzen, qualifizierte Software-Lieferanten für Teilprodukte auszuwählen. Um eine effektive Nutzung der Unterauftragnehmer zu gewährleisten, ist es deshalb notwendig, Vereinbarungen mit den Lieferanten abzuschließen, deren Leistungsfähigkeit und Ergebnisse zu Verfolgen bzw. zu Lenken und in entsprechenden Dokumenten bewertend zu sichern.

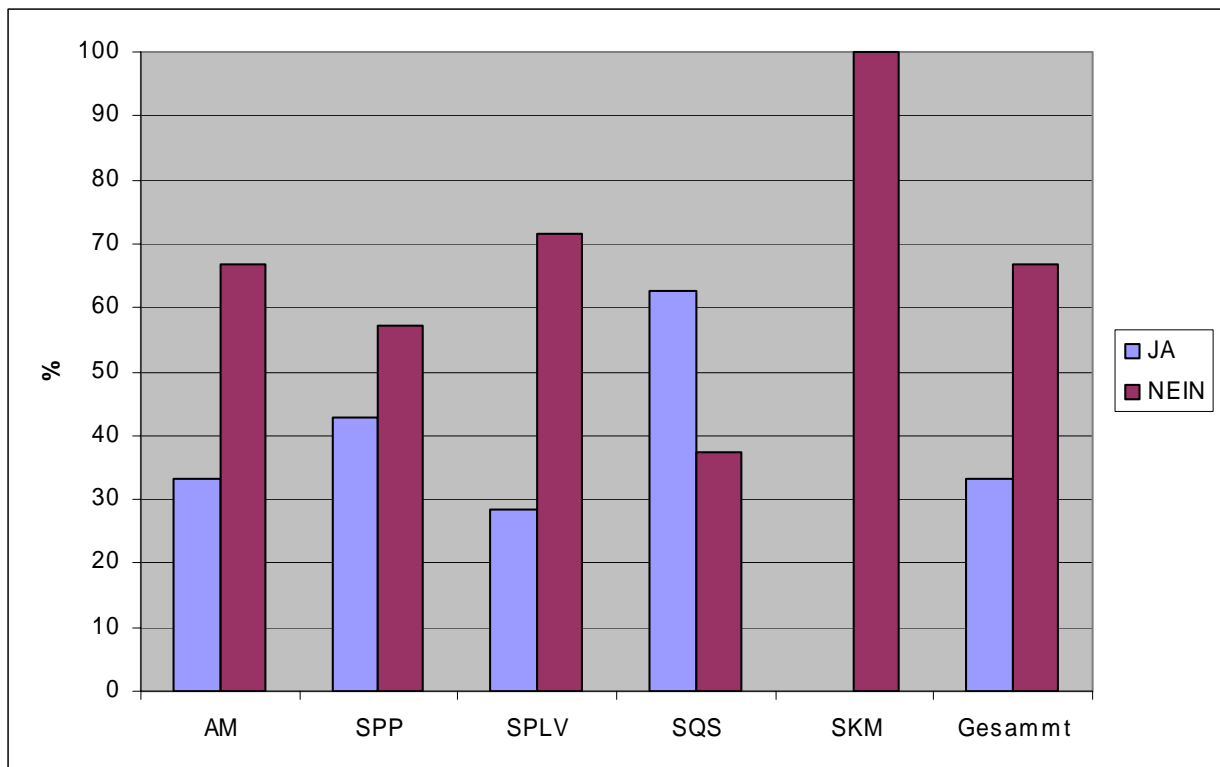
Da die icubic AG keine externe Software akquiriert bzw. Unterprojekte delegiert, verfügt sie nicht über Unterauftragnehmer. Dieser Umstand macht eine Überprüfung des Software-Unterauftragsnehmer-Managements zum gegenwärtigen Zeitpunkt unnötig, da für diesen Abschnitt des CMM-Modells keine Prozessdefinitionen benötigt werden.

## 4. Bewertungsergebnisse

Die aus der Basisbewertung gezogenen Ergebnisse lassen sich in folgender Tabelle darstellen:

	AM	SPP	SPLV	SQS	SKM	Gesamt
JA	2	3	2	5	0	12
NEIN	4	4	5	3	8	24

In Prozentangaben lassen sich diese Werte in folgendes Diagramm überführen:



Die Basisbewertung hat gezeigt, dass die icubic AG nur eine unzureichende Implementierung der Schlüsselprozesse der CMM-Ebene 2 aufzeigt. Der daraus resultierende Handlungsbedarf bildet die Basis der folgenden Abschnitte. Es wird versucht, die fehlenden oder unzureichend implementierten Anforderungen der einzelnen Schlüsselprozesse aufzuzeigen und somit die Basis für deren Einführung bzw. Erweiterung zu legen.

## 4.1. Anforderungsmanagement (AM)

Einer der wesentlichen Punkte des Anforderungsmanagements ist die schriftliche Festsetzung der Erstellungs- und Verwaltungsprozesse für Systemanforderungen mit Software-Bezug. Dieser Punkt wird bei der icubic AG nur unzureichend erfüllt, da sowohl die schriftlich niedergelegten Prozessvorgaben als auch die kontinuierliche Einhaltung dieser Anforderung nicht gewährleistet sind. Zur Erfüllung dieses Punktes ist es notwendig, die bereits sporadisch in mündlicher Form durchgeführten Erstellungs- und Verwaltungsprozesse in schriftliche Form umzuwandeln und auf deren konsequente Einhaltung mittels schriftlicher Vereinbarungen zu bestehen.

Ein weiterer kaum berücksichtigter Punkt dieses Schlüsselprozesses sind die bedarfsspezifischen Schulungen der mit dem Systemanforderungsmanagement betrauten Mitarbeiter. Da derartige Schulungen nur in ungenügendem Maße und diskontinuierlich angeboten werden, ist anzuraten, einen Plan aufzustellen, der diesen Schulungsbedarf abdeckt. Weiterhin sind die Ergebnisse dieser Fortbildungen zu protokollieren und bei späteren Personal- bzw. Schulungsplanungen zu berücksichtigen.

Durch eine Auswertung des Status der Anforderungsmanagementtätigkeiten kann ein Überblick über den Verlauf der Entwicklungen gegeben werden. Diese statistischen

Auswertungen bildet die Grundlage zur Erfüllung eines weiteren ungenügend implementierten Punktes dieses Schlüsselprozesses.

Der letzte unzureichende bzw. falsch implementierte Punkt des Anforderungs-Managements befasst sich mit Reviews der Tätigkeiten des Managements der Systemanforderungen. Diese müssen regelmäßig durch die Software-Qualitätssicherung durchgeführt und überwacht werden. Da diese Reviews aber nur in unregelmäßigen Intervallen durch die Firmenleitung durchgeführt werden, besteht hier Handlungsbedarf. Daher müssen zur Erfüllung dieser Anforderung Zeitpläne erstellt und die Ergebnisse festgehalten werden. Sich daraus ergebende Änderungen der Tätigkeiten des Anforderungsmanagements müssen umgehend umgesetzt werden.

## 4.2. Software-Projektplanung (SPP)

Für die effektive Durchführung der Software-Projektplanung, ist es unabdingbar, dass Schätzungen von zum Beispiel Größe, Kosten oder Zeit dokumentiert und bei der weiteren Planung und Verfolgung des Software-Projektes Verwendung finden. Da zwar grobe Planungen vorgenommen werden, die im späteren Verlauf allerdings keine weitere Verwendung finden, fehlt hier ein entscheidendes Instrument zur Planungsprozess-Verbesserung. Deshalb müssen bei der icubic AG Prozesse eingeführt werden, die die Schätzungen in den fortlaufenden Planungs- und Steuerungsablauf einbinden.

Ähnlich den Schätzungen muss auch bei weiteren Planungstätigkeiten verfahren werden. Das beinhaltet die schriftlich dokumentierte Vorgabe von Prozessdefinitionen zur Software-Projektplanung auf Basis der bereits eingesetzten Projektplanungs-Schemata, sowie die Notwendigkeit, die zur Software-Projektplanung benötigten Ressourcen zu planen und in ausreichendem Maße zur Verfügung zu stellen.

Der abschließende nicht erfüllte Punkt dieses Schlüsselprozesses beschäftigt sich mit der mangelnden Statusbewertung bzw. Messung der Planungstätigkeiten. Die Statusbewertungen bei Auslieferung des Software-Produktes geben nur ungenügend Auskunft über den Planungsprozess. Um eine höhere Lenkbarkeit dieses Prozesses zu bewirken, ist es notwendig, in der icubic AG Meilensteine festzulegen, die während der laufenden Projektplanung Messungen des Status ermöglichen.

## 4.3. Software-Projektlenkung und –Verfolgung (SPLV)

Da nur sehr geringfügig Schätzungen während der Planungsphase durchgeführt werden, ist ein wirkungsvoller Vergleich mit den tatsächlich erzielten Ergebnissen nicht möglich. Dieser Mangel an wirkungsvollen Vergleichsmöglichkeiten lässt sich in erster Linie auf das Fehlen von schriftlichen Vorgaben zur Verfolgung und Steuerung der Software-Entwicklungstätigkeiten zurückführen. Diese Vorgaben bilden die Grundlage der Software-Projektlenkung und –Verfolgung. Auf Grund dessen ist das Hauptaugenmerk auf deren schriftliche Festlegung zu richten. Wie in den anderen Managementformen müssen auch hier Vereinbarungen mit den betroffenen Personen und Personengruppen getroffen und ihre konsequente Einhaltung sichergestellt werden.

Wie schon bei den Vereinbarungen müssen auch die Verantwortlichkeiten innerhalb der Projekte festgelegt werden. Die Verantwortlichkeiten innerhalb der

Software-Projektlenkung und –Verfolgung beziehen sich auf die Verfolgung der Fertigstellung von Arbeitsergebnissen und der Tätigkeiten. Da diese Verantwortlichkeiten bei der icubic AG nicht genau festgelegt sind, bilden sie einen Risikofaktor bei der Software-Projektlenkung und –Verfolgung.

Um einen reibungslosen Ablauf der Lenkungs- und Verfolgungstätigkeiten zu gewährleisten, ist es notwendig, den Status dieser Tätigkeiten durch Messungen festzustellen und zu analysieren. Die Ergebnisse dieser Bewertungen müssen dem Projektinformationsfluss wieder zugeführt werden. Werden die Verfolgungs- und Steuerungstätigkeiten des Software-Projektes periodischen Reviews durch das übergeordnete Management unterzogen, kann somit ein ständiger Verbesserungsprozess vorangetrieben werden. Momentan werden diese Reviews bei der icubic AG allerdings nur unzureichend durchgeführt, was einen Handlungsbedarf im Sinne des CMM impliziert.

#### 4.4. Software-Qualitätssicherung (SQS)

Wie bereits zuvor erwähnt werden in der icubic AG Reviews und Audits der Software-Qualitätssicherung nur in unzureichendem Maße durchgeführt. Durch eine unzureichende Übermittlung der Ergebnisse an die beteiligten Personen und Personengruppen wird dieser Mangel noch zusätzlich verstärkt. Dieser Umstand bewirkt einen Einflussverlust der Software-Qualitätssicherung auf den laufenden Entwicklungsprozess. Um dieser Entwicklung entgegenzuwirken ist es notwendig, sowohl projektspezifische als auch -übergreifende Reviews und Audits in zeitlich festgesetzten Abständen durchzuführen und Verfahren zur Weiterleitung der Ergebnisse zu implementieren. An die übermittelten Ergebnisse müssen Vereinbarungen mit den beteiligten Personen und Personengruppen geknüpft werden, die deren Umsetzung sicherstellen.

Wie bereits bei den anderen Managementformen besteht auch in diesem Schlüsselprozess bei der icubic AG großer Handlungsbedarf bei der Erstellung schriftlich niedergelegter Vorgaben zur Durchführung ihrer Aktivitäten. Durch die Festlegung solcher Verfahren kann die weitgehend außerhalb der Projekte angesiedelte Qualitätssicherung ihren Einfluss auf den gesamten Entwicklungsprozess erweitern und somit die Steigerung und Überwachung der Produktqualität gewährleisten.

Ein weiteres Mittel zur Steigerung der Prozess- und Produktqualität besteht in der Erhöhung der Effektivität der Software-Qualitätssicherungstätigkeiten. Diese Effizienzsteigerung lässt sich durch eine Bewertung der bereits erhobenen Messdaten wie Kosten und Status der Aktivitäten bewirken. Bei der icubic AG werden solche Messdaten erhoben, aber nicht bewertet und können somit nicht wieder in den Projektfluss eingefügt werden.

#### 4.5. Software-Konfigurations-Management (SKM)

Durch das Fehlen des Software-Konfigurations-Managements ergeben sich Anforderungen ähnlich denen der anderen Schlüsselprozesse. Diese beinhalten die Planung der Software-Konfigurations-Managementtätigkeiten und das Erstellen von Verfahren zur Identifizierung der Arbeitsergebnisse des Software-Konfigurations-Managements. Weiterhin müssen Verfahren zum Durchführen von geordneten

Änderungen an diesen Erwartungen und für deren Bereitstellung erstellt werden. Weitere schriftlich niedergelegte Vorgaben müssen für die Durchführung der Software-Konfigurations-Managementtätigkeiten und für Verfahren zur Änderungskontrolle eingeführt werden. Statusmessungen der Software-Konfigurations-Managementtätigkeiten und periodische Audits sind zeitlich festzulegen und durchzuführen. Die Berichte dieser Überprüfungen sind an die betroffenen Personen und Personengruppen weiterzuleiten. Nicht zuletzt müssen die mit den Software-Konfigurations-Managementtätigkeiten beauftragten Mitarbeiter und Manager in ihrem Arbeitsbereichen entsprechend den Anforderungen geschult und Erweiterungen ihrer Kenntnisse vermerkt werden.

Da das Software-Konfigurations-Management als Managementform innerhalb der icubic AG so nicht vorhanden ist, muss die konkrete Implementierung dieses Schlüsselprozesses zukünftig intensiv diskutiert werden. Aus diesem Grund können hier nur die wesentlichen Anforderungen an diesen Schlüsselprozess nur skizziert werden.

Die Umsetzung dieser Anforderungen stellt für die icubic AG einen grundlegenden Schritt zum Erreichen der CMM-Stufe 2 dar.

## **5. Fazit**

Nach dem gegenwärtigen Stand hat die icubic AG das gewünschte Ziel des CMM Level 2 nicht erreicht. Unserer Bewertung nach befindet sich das Unternehmen derzeit auf dem nicht definierten Level 1.3. Da allerdings einige wichtige Anforderungen der sechs Schlüsselbereiche teilweise bereits implementiert sind, ist eine Korrektur nach oben auf 1.5 durchaus vertretbar. Die Bewertungsergebnisse können dabei als Basis für eine CMM konforme Weiterentwicklung angesehen werden.

## 9 Referenzen

[Ahern2003] Ahern, Dennis M., Clouse Aaron, Turner Richard: CMMI Distilled: A Practical Introduction to Integrated Process Improvement, Second Edition. Addison Wesley. 2003

[Mutafelija 2003] Mutafelija, B. und Stromberg, H.: Systematic Process Improvement Using ISO 9001:2000 and CMMI<sup>®</sup>. Artech House, Inc. Norwood. 2003.

[Zuse 2003] Zuse, H. und Drabe, K.: ZD-MIS: Zuse/Drabe-Measure-Information-System. 1993 bis 2003. Erhältlich bei: <http://www.zuse.info>.

[Balzert 2000] Balzert, H.: Lehrbuch der Software-Entwicklung. Spektrum Akademischer Verlag. 2000

[nach DumkeSE2] Webservice: Vorlesungsscript Software-Engineering I: <http://ivs.cs.uni-magdeburg.de/~dumke/ST2/ST2Einf.html>

[DIN8402] DIN EN ISO 8402: Quality management and quality assurance – Vocabulary

[ZusePaper] Zuse, H.: Messung von Softwarequalität

[Wiki1] Webservice: Wikipedia–Die freie Enzyklopädie: [http://de.wikipedia.org/wiki/Metrik\\_%28Software%29](http://de.wikipedia.org/wiki/Metrik_%28Software%29)

[Kan2003] Kan, Stephen H.: Metrics and Models in Software Quality Engineering, Second Edition. Addison Wesley. 2003

[Crosby1979] Crosby, P. B.: Quality is Free: The Art of Making Quality Certain. McGraw-Hill. 1979

[Juran1970] Juran, J. M. and Gryna, F. M. Jr.: Quality Planning and Analysis: From Product Development Through Use. McGraw-Hill. 1970

[Fenton1997] Fenton, Norman E., Pfleeger, Shari Lawrence: Software Metrics: A Rigerous & Practical Approach. International Thomson Computer Press. 1997

[Neumann03] Neumann Robert, Messen von Softwarekomplexität, Seminararbeit, 2003

[Geocity1] Webservice: Funktional Size Measurement: [http://www.geocities.com/lbu\\_measure/fpa/fpa.htm](http://www.geocities.com/lbu_measure/fpa/fpa.htm)

[Lother1] Paper: Lother, M., Dumke R.: POINTS METRICS – Comparison and Analysis. University of Magdeburg.

[Kulpa03] Margaret K. Kulpa, Kent A. Johnson: Interpreting the CMMI<sup>®</sup> - A Process Improvement Approach. Auerbach Publications. 2003.

[Wang2000] Wang, Yingxu; King, Graham: Software Engineering Processes – Principles and Applications. CRC Press LLC. 2000.

[Kneuper2003] Kneuper, Ralf: CMMI – Verbesserung von Softwareprozessen mit Capability Maturity Model Integration. dpunkt.verlag GmbH. 2003.

[Rook1991] Rook, Paul: Software Reliability Handbook. Elsevier Applied Science - London and New York. 1991

[WU-Wien] Webservice: Wirtschafts-Universität Wien: <http://wwwai.wu-wien.ac.at/~hahsler/research/diss/diss/node5.html#vorgehen>

[HAW-Hamburg] Webservice: Hochschule für Angewandte Wissenschaften: <http://users.informatik.haw-hamburg.de/~khh/st4se2/node8.html>

[Boehm1976] Boehm, Barry W. and J. R. Brown, M. Lipow: Quantitative Evaluation of Software Quality. in: Proceedings of the International Conference on Software Engineering. 1976.

[McCall1977] McCall, James A. and P. K. Richards, G. F. Walters: Factors in Software Quality, Volume I: Concepts and Definitions of Software Quality. 1977.

[Wallmüller2001] Wallmüller, Ernest: Softwarequalitätsmanagement in der Praxis. Hanser Verlag. München/Wien. 2001.

[CMMI01] Gruppe 8. Capability Maturity Model Integration. Vortrag zu Wirtschaftsinformatik (Mo\_Vortrag\_CMM\_final.pdf).

[Quagmire2005] Webservice: Software Productivity Consortium: [www.software.org/quagmire](http://www.software.org/quagmire)