

# CMMM

## *Projektmanagement bei Softwareprojekten*

von

**Steffen Dettmer** und **Martin Linke**

**1.Juli 2000**

## **Inhalt**

Inhalt .....	1
Einleitung .....	3
Kunst .....	3
Capability Maturity Model (CMM) .....	4
Level der Reife .....	4
1. Initial .....	4
2. Repeatable .....	4
3. Defined .....	5
4. Managed .....	5
5. Optimized .....	5
Key Process Areas .....	5
Definition .....	5
Ziele .....	5
Voraussetzungen .....	6
Ausgangspunkt Praxis .....	7
Fehlendes Mangement .....	7
Fehlende Einsicht .....	7
Entwicklung der Entwicklung .....	8
Nachvollziehbarkeit .....	8
Genauere Definitionen .....	9
Management der Entwicklung .....	9
Ständige Verbesserungen .....	10
Sparen durch Mehraufwand .....	10
Die Rolle des Projekt Managements .....	11
Chaos herrscht in drei von vier Unternehmen .....	11
Die reale Situation .....	11
Langwierige Entwicklung .....	12
Vorteile in der Praxis .....	12
Feststellen des Niveaus .....	13
Evaluation durch SEI .....	13
Einsatz .....	14
Zusammenfassung des Capability Maturity Model .....	14
Probleme und Grenzen .....	15

## **Einleitung**

Beim Realisieren von Softwareprojekten sind einige spezielle Probleme zu beachten. Soll eine Software erstellt werden, so gibt es zwar eine Vielzahl von Ideen, die eine hohe Qualität und Zuverlässigkeit sichern, leider werden diese zu selten angewendet. Ein Softwareentwickler ist zu oft nur ein Programmierer; er denkt, das Schreiben von Programmcode sei seine Hauptaufgabe. Die eigentliche Entwicklung wird dabei dann vernachlässigt, Probleme sind vorprogrammiert, im wahrsten Sinne des Wortes. Hier ist ein Umdenken erforderlich. Die Grundlagen des Softwareengineering müssen nicht nur bekannt sein, sondern auch angewendet werden. Die konsequente Durchsetzung dieser Prinzipien liegt in der Verantwortung des Projekt Managers, aber die Entwickler müssen dies auch in der Praxis umsetzen. Dies zu erlernen, ist nicht einfach, und erfordert Zeit. Es ist im Laufe eines Projektes wohl nicht möglich, aus einem Drauflos-Programmierer einen guten Software Ingenieur zu machen. Es ist mehr ein Prozeß, der nie endet, und in dessen Verlauf die Entwickler immer besser werden, sorgfältiger arbeiten und dokumentieren. Leider wird in der Praxis oft zu wenig Wert darauf gelegt. Die Folge sind niedrige Softwarequalität, schlechte Wartbarkeit, schlechte Aufwandsschätzungen und hohe Fehlerquoten; Termine können nicht eingehalten werden, Kunden sind unzufrieden und einmal erstellte Programme können später nur mit großem Aufwand erweitert werden.

## **Kunst**

Ein unerfahrener Programmierer schreibt schnellen Code. Nur er versteht ihn, und daß auch nur, wenn er ihn schreibt; später auch er selbst nicht mehr. Die Entwicklungszeit besteht zu mehr als 90% aus Implementation. Es gibt kaum begleitende Unterlagen, keine Dokumente, die die Strukturen und Algorithmen erklären. Man kann vielleicht einzelne Programmierer an ihrer „Handschrift“ erkennen. Das Optimieren des Codes wird nicht dem Compiler überlassen, wilde Konstruktionen sind die Folge. Funktionalität hat höhere Priorität als Stabilität. Das Programm wird zum Kunstwerk. Der Programmierer drückt sich selbst aus, unverständlich für andere. Ist das Werk fertig, das heißt, funktionieren einige Teile, so ist ein Abschluß da. Später kann das Kunstwerk nicht mehr an kleinen Stellen geändert werden. Erfüllt es einige Anforderungen nicht, so muß ein vollständig neues Werk geschaffen werden. Der Code enthält keine Kommentare, es gibt kaum Schnittstellen und kein Pflichtenheft. Das Chaos wird von einigen Genies beherrscht, die stundenlange Funktion um Funktion in einen komplexen, unübersichtlichen Code einhacken. Entwicklungszeiten und –kosten werden aus dem Bauch geschätzt. Schnittstellen werden erst entwickelt, wenn man

merkt, daß man auf Module eines anderen zugreifen muß. Werden die Projektteile zusammengefügt, funktioniert überhaupt nichts, aber niemand fühlt sich verantwortlich. Ein Handbuch entsteht nebenbei nach der Implementation, an Hand des fertigen Produktes. Niemand denkt an spätere Wartbarkeit. Kaum getestet wird geliefert. Leicht wird der Einsatz beim Kunden zu einem Desaster, oder das Produkt wird nie fertig, da die Komplexität die Produktivität überschreitet.

## **Capability Maturity Model (CMM)**

Um derartige Probleme in den Griff zu bekommen, finanziert das US-Verteidigungsministerium seit 1984 das Software Engineering Institute der Carnegie Mellon University. Von dort stammt ein in den USA beliebtes Modell, mit dessen Hilfe sich die Organisation und somit - vielleicht - die Qualität eines Software-Unternehmens einstufen läßt. Das CMM für Software beschreibt die Prinzipien und Praktiken, die der Software – Prozeßreife zugrunde liegen. Es soll Softwareunternehmen bei der Verbesserung ihrer Prozesse von chaotischen hin zu disziplinierten Softwareprozessen unterstützen. Dieses „capability maturity model“ (CMM) klassifiziert die Reife einer Softwareabteilung oder eines Software-Unternehmens als CMM-Level 1 bis 5.

### **Level der Reife**

Die Levels sind definiert durch 18 sog. 'key process areas' (KPA), die die Projektperformance der Levels 2 bis 5 charakterisieren.

Die fünf Levels des CMM:

#### 1. Initial

Der Softwareprozeß ist gekennzeichnet als 'ad hoc' und häufig chaotisch. Wenige Aufgaben sind definiert. Der persönliche Einsatz einzelner Entwickler ist für den Erfolg hauptverantwortlich.

#### 2. Repeatable

Grundlegende Projektmanagement-Prozesse hinsichtlich Kosten-, Zeit- und Funktionsplanung sind festgelegt. Abläufe werden nachvollziehbar, auf Erfahrungen kann zurückgegriffen werden.

### 3. Defined

Abläufe für Management, Entwicklung und Engineering sind dokumentiert, standardisiert und integriert. Es gibt Standards für die Organisation der Entwicklung und Wartung. Die Schritte sind genau beschrieben.

### 4. Managed

Abläufe werden kontrolliert. Es kann steuernd von zentraler Stelle eingegriffen werden, da die Entwicklung transparent ist, und gut dokumentiert wird.

### 5. Optimized

Erfahrungen werden wiederverwendet. Dazu werden diese von zentraler Stelle zusammen mit neuen Ideen durchgesetzt. Es wird der Ablauf analysiert, um Schwachstellen zu finden, und diese können ausgemerzt werden.

## ***Key Process Areas***

### Definition

Ein Schlüsselprozeß ist ein Prozeß, der zum Erreichen eines höheren CMM Levels beiträgt. Anhand dessen kann ein Unternehmen auf eine bestimmte CMM Stufe klassifiziert werden.

Ein Bereich enthält Ziele, durch deren Erreichen die Produktivität erhöht wird. Das konkrete Aussehen dieser Ziele hängt vom Projekt und vom Unternehmen ab.

### Ziele

Um die Prozeßfähigkeit und -reife einer Organisation, und damit deren Leistungsfähigkeit auf eine höhere Stufe zu bringen, müssen bestimmte Schlüsselprozesse (Key Process Areas) etabliert werden, diese kann man in Bereichen zusammenfassen. Diese Bereiche zeigen auf, wo eine Organisation Abläufe optimieren kann. Um eine CMM Stufe zu erreichen, müssen alle Ziele dieser Schlüsselprozesse vollständig erreicht sein. Diese Prozesse sind hier aufgeführt und werden nachstehen genauer erklärt.

Level 2	Level 3	Level 4	Level 5
Repeatable	Defined	Managed	Optimized
<ul style="list-style-type: none"> <li>• Management der Anforderungen</li> <li>• Software Projekt Management</li> <li>• Management von Unter – Auftragnehmern</li> <li>• Qualitätssicherung</li> <li>• Konfigurations-Management</li> </ul>	<ul style="list-style-type: none"> <li>• Unternehmensweite Prozeßorientierung</li> <li>• Unternehmensweite Prozeßdefinition</li> <li>• Training/Schulung</li> <li>• Software Management integriert</li> <li>• Produkt Engineering</li> <li>• Koordination über Gruppengrenzen</li> <li>• Rezensionen</li> </ul>	<ul style="list-style-type: none"> <li>• Qualitäts Management</li> <li>• Erweitertes Prozeß-Management</li> </ul>	<ul style="list-style-type: none"> <li>• Problemerkennung und Verhinderung</li> <li>• Reaktion auf neue Technologien und deren Integration</li> <li>• Änderungen der Abläufe bei Bedarf</li> </ul>

### Voraussetzungen

#### KPAs des Levels 2:

Requirements Management (RM)  
 Software Project Planning (PP)  
 Software Project Tracking and Oversight (PT)  
 Software Subcontract Management (SM)  
 Software Quality Assurance (QA)  
 Software Configuration Management (CM)

#### KPAs des Levels 3:

Organization Process Focus (PF)  
 Organization Process Definition (PD)  
 Training Program (TP)  
 Integrated Software Management (IM)  
 Software Product Engineering (PE)  
 Intergroup Coordination (IC)  
 Peer Reviews (PR)

#### KPAs des Levels 4:

Quantitative Process Management (QP)  
 Software Quality Management (QM)

#### KPAs des Levels 5:

Defect Prevention (DP)  
Technology Change Management (TM)  
Process Change Management (PC)

Alle 18 KPAs sind strukturiert in 5 “Key Practices”:

Commitment to Perform (CO)  
Ability to Perform (AB)  
Activities Performed (AC)  
Measurement and Analysis (ME)  
Verifying Implementation (VE)

## ***Ausgangspunkt Praxis***

### Fehlendes Management

Die „nullte“ und die erste Stufe könnte man weniger höflich als Chaos bezeichnen. Es herrscht Mangel an Planung: Von Tag zu Tag ändern sich spontan die Ziele, so fern es überhaupt welche gibt; es gibt keine Vorgaben für die Entwickler in Form von Normen oder Standards, das Management der Software-Entwicklung fehlt ganz oder ist nur rudimentär vorhanden. Die Unternehmensleitung wird mangelhaft oder gar nicht und nicht konkret und nachvollziehbar über den Fortschritt von Projekten unterrichtet – nicht einmal der Projektleitung ist der Stand bzw. Fortschritt genau bekannt. Trotzdem können Unternehmen mit dieser Art gute Software entwickeln - allerdings nicht als Folge eines definierten Prozesses, sondern durch die Anstrengungen und das Können Einzelner oder kleiner Teams, Erfolg kann nicht geplant oder garantiert werden. Verlassen diese Mitarbeiter die Firma, so ist der Erfolg nicht wiederholbar, die Organisation fällt im Wettbewerb zurück. Spätere Wartung ist ohne diese Mitarbeiter kaum möglich. Also können diese Stufen des CMM für das Management nicht wünschenswert sein.

### Fehlende Einsicht

Das oben beschriebene Team bewegt sich weit entfernt von Reife, es bewegt sich sozusagen auf CMM Level 0. Man sieht jedoch einige Fehler, und lernt aus diesen. Neue Regeln werden aufgestellt: Man muß erst die Lösung für das Problem kennen und verstanden haben, erst dann darf programmiert werden. Schnittstellen werden im voraus diskutiert und festgelegt. Da die

Leute immer noch die selben sind, ändert sich in der Praxis nicht allzuviel. Die neuen Methoden sind ungewohnt, das Entwickeln dauert länger. Niemand ist so recht vom Sinn des Mehraufwandes überzeugt, und so wird weitergearbeitet, wie bisher. Man bewegt sich auf CMM-Level 1 („initial“) zu. Direkte Vorteile sieht aber erst einmal niemand.

## ***Entwicklung der Entwicklung***

### Nachvollziehbarkeit

Im Gegensatz zur Stufe 1 kann der erzielte Erfolg auf der mit "repeatable" bezeichneten zweiten Stufe wiederholt werden.

Dem Projekt-Management ist es möglich, einzelne Entwicklungsschritte der Software zu verfolgen; ein Zeitplan wird aufgestellt und seine Einhaltung überwacht. Der Fluß personeller Aufwände und finanzieller Mittel läßt sich verfolgen, weshalb das Management bei Abweichungen steuernd eingreifen kann. Laufende Projekte können mit anderen laufenden und abgeschlossenen Projekten verglichen werden. So ist es möglich, aus anderen Entwicklungsprojekten des eigenen Unternehmens zu lernen.

CMM-Level 2 („repeatable“) kann erreicht werden, wenn nun weiter die richtigen Konsequenzen gezogen werden. Die Entwicklung soll nachvollziehbar, wiederholbar werden. Schulungen werden durchgeführt. Die Entwickler lernen die Grundlagen des Softwareengineering kennen. Man hört von Analyse und Design. Die Entwickler halten sich oft viel zu wenige daran, Arbeitstil ändert sich eben nicht von heute auf morgen. Man verwendet zwar neue Methoden, aber nur, wenn es die Arbeit nicht stört. Kommt es hart auf hart, wird nur gehackt, und die Probleme bleiben. Neben den Schulungen wird auch die Abteilung strukturiert. Plötzlich gibt es Leute, die sich mit dem Testen beschäftigen, und ständig die Programmierer wegen ihres Codes kritisieren, aber selbst nichts programmieren. Qualitätssicherung nennt man das, aber der Sinn wird den Entwicklern nicht richtig klar, die Spezifikationen für die Tests werden nebenbei mal eben geschrieben.. Die Entwickler fühlen sich unter Druck gesetzt. Inzwischen merken die Projektmanager Veränderungen, so wird der Aufwand realistischer geschätzt, aber der Durchbruch bleibt aus. Das Modell funktioniert nicht optimal. Jetzt müssen die Schwachstellen gesucht und beseitigt werden. Die *Entwicklung selbst* muß kritisch betrachtet und optimiert werden.

## Genaue Definitionen

Nun wird man erkennen, daß etliche Festlegungen fehlen bzw. nicht eingehalten werden. Nun werden auch Arbeitsabläufe definiert und Protokolle erstellt. Analyse und Design werden überwacht, und können nicht mehr übersprungen werden. Es sieht zwar nun etwas nach einer Behörde aus, aber die Richtlinien werden durchgesetzt. Es gibt ein großes Regelwerk, daß einzuhalten ist. Die eigentliche Implementation ist nur noch ein kleiner Bestandteil. Es gibt ein Pflichtenheft und viele begleitende Dokumente. Tests gehören zu jeder Phase, Fehler werden früher erkannt, und sind schneller behebbar. Die Entwickler merken, daß Terminverzug seltener wird, und weniger Überstunden anfallen. Man spricht von CMM-Level 3.

Bei den Unternehmen, deren Entwicklungsprozeß der, mit "defined" gekennzeichneten Stufe des Modells zuzuordnen ist, basiert der Entwicklungsprozeß auf Normen und Standards, die für die gesamte Organisation und alle Projekte Gültigkeit haben. Projektspezifisch werden detaillierte Pläne für die Bereiche Entwicklung, Qualitätssicherung und Konfigurationsmanagement erstellt. Die Einhaltung dieser Vorgaben wird überwacht.

## ***Management der Entwicklung***

Endlich läuft die Entwicklungsabteilung. Den Projektmanager interessiert aber, wie effizient gearbeitet wird, wo Zeit gespart werden kann, wo häufige Fehler vermieden werden können. Es werden zusätzlich zur Qualitätssicherung weitere Dinge getestet, analysiert und statistisch ausgewertet. Wieviel Zeilen Code schreibt ein Programmierer? Wieviel Fehler werden darin entdeckt? Wie kann man diese Quote verbessern? Sind genügend viele Kommentare enthalten? Wie komplex sind die Funktionen? Um das untersuchen zu können, kommen Metriken zum Einsatz. Spezielle Programme untersuchen den Code nach verschiedenen Gesichtspunkten. Der Entwicklungsprozeß wird unter die Lupe genommen, an vielen Stellen wird korrigierend eingegriffen. Management must manage. Projektleiter haben endlich harte Fakten in der Hand, können genau planen. Die Entwicklung wird effizienter. Damit ist CMM-Level 4 („managed“) erreicht.

Auf der Stufe 4 kommt eine quantitative Bewertung hinzu. Dabei werden sowohl der Entwicklungsprozeß als auch die entstehenden Softwareprodukte durch Messungen überwacht. Metriken sind die geeigneten Werkzeuge, um eine fundierte Aussage zur Software und deren Entstehung machen zu können.

## **Ständige Verbesserungen**

Nun werden Probleme früher bzw. überhaupt erst einmal erkannt. Damit können die Ursachen gefunden und beseitigt werden. Die Entwicklungsphase ist in Analyse, Design, Implementation, Test, Inbetriebnahme und Wartung unterteilt (im Zusammenhang mit CMM konzentriert man sich dabei auf die ersten vier Punkte). Man leistet richtiges Softwareengineering. Der Entwicklungsprozeß als solcher ist übersichtlich und planbar. Die Kommunikation untereinander nimmt einen wichtigen Part ein, durch die Abhängigkeiten der verschiedenen Phasen ist es nicht anders möglich. Viele Leute arbeiten an einem Projekt. Zeitpläne werden eingehalten. Gibt es Störungen, so fallen diese auf, dessen Ursachen werden beseitigt. Man vermeidet derartige Probleme beim nächsten Mal. Man erkennt, wo zuviel Zeit oder andere Ressourcen verbraucht werden, und kann steuernd eingreifen. Die Entwicklungsabteilung wird ständig optimiert. Man erreicht CMM Level 5.

Diese fünfte und (vorläufig?) letzte Stufe des Capability Maturity Models wird als "optimized" bezeichnet. Hier lassen sich die Messungen der vorhergehenden Stufe nutzen, um Schwachpunkte zu identifizieren und Verbesserungen einzuführen. Neue Ideen und Technologien können übernommen und in der Praxis erprobt werden.

## **Sparen durch Mehraufwand**

Auf den ersten Blick sieht man einigen Mehraufwand. Viel Papier wird bedruckt, viele Leute beschäftigt, die eigentlich gar nicht mehr programmieren. Die eigentliche Implementation macht je nach Projekt nur noch 20%-30% des Projektumfanges aus. Und doch ist es der richtige Weg. Es ist eben nicht möglich, die restlichen 70% wegzulassen, denn in diesem Fall dauert die Implementation zehnmal so lange. Dabei entsteht dann auch noch nur unzureichende Qualität – von der fehlenden Wartbarkeit ganz zu schweigen.

Wenn ein Programmierer aus dem Kopf drauflos hackt, muß er nebenbei ja den selben Analyse- und Designaufwand treiben, als wenn diese Tätigkeiten in einer eigenen Phase ablaufen würden. Nur entstehen so Nachteile, da Teile des Codes nicht zu neuen Designentscheidungen passen, Analysen unvollständig bleiben, weil Teile vergessen werden und nicht nachvollziehbar sind.

Ein Grundprinzip heißt „Teile und Herrsche“, eigentlich ist dieses Phasenmodell ja nur eine Zerlegung in beherrschbare Teile. Stellt man beim Design bereits ein Problem oder einen Fehler fest, ist er leicht korrigierbar. Stellt man bei der Implementation oder schlimmer – beim Test – einen

Design- oder Analysefehler fest, ist der Korrekturaufwand immens. Es kann vorkommen, daß große Teile des Codes neugeschrieben werden müssen, möchte man einen hohen Qualitätsstand erreichen. Hinzukommend sind große Projekte einfach nicht anders beherrschbar, da das menschliche Gehirn nur endliche Kapazität hat. Die Struktur von einer Million Zeilen Code kann niemand mehr komplett im Kopf haben, und nach einem Jahr noch richtig anwenden, wenn kein entsprechendes Dokument vorhanden ist.

## ***Die Rolle des Projekt Managements***

Nun stellt sich die Frage, wie das Projekt Management von diesen Erkenntnissen profitieren kann. Macht man es von Anfang an richtig, zu lassen sich viele Probleme vermeiden. Besonders gute Chancen hat man, wenn ein neues Team zusammengestellt wird, da keine tiefsitzenden, festen Arbeitsabläufe existieren, und sich soziale Kommunikationswege erst bilden. Ein bestehendes Team zu gutem Softwareengineering zu bringen, also den CMM-Level zu erhöhen, ist ein langwieriger, komplizierter Prozeß. Dabei sind auch gute soziale, emotionale bzw. psychologische Fähigkeiten der Projektleitung von großer Bedeutung. Mit Gewalt lassen sich neue Methoden kaum durchsetzen; verstehen und erkennen die Entwickler jedoch die Vorteile, so wird es viel einfacher für das Management. Es muß ständig daran gearbeitet werden, sauber und strukturiert zu arbeiten und Fehler zu vermeiden. Es darf nicht ein bestimmter CMM Level als Ziel anvisiert werden, dieser ergibt sich vielmehr, wenn die Maßnahmen des Management Erfolge zeichnen und Früchte tragen. Es muß Rückkopplungen zwischen Leitung und Ausführung geben. Die Grundlage dafür heißt Kommunikation. Ineffizienz kann nur vermieden werden, wenn klar ist, was machbar ist, wo Probleme entstehen, und die Entwicklung nachvollziehbar ist. Die erste Aufgabe ist also, den Entwicklungsprozeß selbst zu erkennen, zu verstehen und zu begreifen. Schließlich kann erst dann überhaupt sinnvoll und zielgerichtet eingegriffen werden.

## ***Chaos herrscht in drei von vier Unternehmen***

### Die reale Situation

Das „Software Engineering Institute“ hat 1989 eine Untersuchung in der amerikanischen Industrie durchgeführt, um herauszufinden, wie es derzeit um den Reifegrad der Software-Entwicklungsprozesse bestellt ist. Das schockierende Ergebnis: Drei Viertel der befragten Organisationen bringen gute Software –vermutlich nur dadurch zustande, daß sich eine Handvoll

Entwickler die Nächte um die Ohren schlagen, um die Software doch noch wenigstens halbwegs zum Laufen zu bringen.

Selbst renommierte Unternehmen wie „Hughes“ sind derzeit nur auf Stufe „2+“ einzuordnen. Lediglich magere drei Prozent der befragten Unternehmen verfügen über einen definierten Software-Entwicklungsprozeß. Die Ebenen 4 und 5 sind überhaupt nicht besetzt, traut man den Ergebnissen des SEI.

### Langwierige Entwicklung

Nach Schätzung der Spezialisten von „Carnegie-Mellon“ braucht ein Unternehmen zwischen 18 und 36 Monate , um die nächst höhere CMM Stufe zu erreichen. Um eine Organisation der Stufe 5 zu werden, kann man also zehn Jahre ansetzen, wenn man bei Null beginnt, aber die meisten Unternehmen werden maximal ein CMM Level von zwei oder vielleicht drei erreichen können. Es ist ja auch nicht möglich, von Stufe 2 direkt auf Stufe 4 zu springen, da diese aufeinander aufbauen.

### Vorteile in der Praxis

Wie sehen nun die Auswirkungen in der Praxis aus? Wichtig ist hier, die Nutzen zu erkennen, die durch die Verwendung und Optimierung von Prozessen entstehen. Der Hauptvorteil ist der Nutzen für das Management: Das Projekt wird übersichtlich, Teile und Etappen werden erkennbar, Arbeitspakete können definiert und abgearbeitet werden. So kann ein genauer Plan gemacht werden, und Abweichungen werden sichtbar und können korrigiert werden. Damit kann der Zeitplan kontrolliert und eingehalten werden, so daß die Projekte und Produkte zum großen Teil pünktlich fertig gestellt werden können. Im Extremfall wird mindestens rechtzeitig erkannt, daß Ziele nicht erreicht werden können, so daß das Management z.B. zusätzliche Entwickler hinzuziehen kann, oder andere Maßnahmen ergreifen kann.

Das Modell liefert einen Maßstab, um Verbesserungen zu erreichen, Unternehmen, Projekte und Teams vergleichen und beurteilen zu können. Weiterhin kann langfristige Planung kontrolliert werden, Fortschritte können gemessen und bewertet werden. Durch dieses Modell kann der Kunde potentielle Auftragnehmer nach konkreten Kriterien bewerten und beurteilen, die Qualität des Entwicklungsprozesses und damit die Qualität des Softwareproduktes abschätzen und denkbare Risiken erkennen und vergleichen. Weiterhin können verschiedene Anbieter bzw. mögliche Auftragnehmer verglichen werden.

## Feststellen des Niveaus

Dies ist natürlich nur unter der Voraussetzung machbar, daß eine möglichst objektive und treffende Einschätzung bzw. Bewertung des Unternehmens vorhanden ist. Dafür gibt es im Wesentlichen zwei Vorgehensweisen: Das Unternehmen kann sich selbst einschätzen (Assessment), oder von Kunden bzw. Auftraggebern bewertet werden (Evaluation).

Ein Assessment dient dem Nutzen des eigenen Unternehmens. Die Untersuchungsergebnisse bieten dem Management Anhaltspunkte über den Stand des Entwicklungsprozesses und können richtungsweisende Denkanstöße zur Verbesserung geben. Diese Ergebnisse sind in der Regel firmenintern und vertraulich. Sie dienen hauptsächlich, um die eigene Produktivität zu messen und zu erhöhen.

Die Evaluation veranlaßt der Auftraggeber bei Projektdurchführung oder bereits vorher, bei der Wahl des Auftragnehmers. Er erhält dann die Resultate einer entsprechenden Untersuchung, und kann damit Risiken, Möglichkeiten und die Produktivität der Prozesse und damit die erwartete Softwarequalität beurteilen. Dadurch kann der Kunde die Einhaltung des Vertrages und des Zeitplanes überwachen und mögliche Probleme frühzeitig erkennen.

## Evaluation durch SEI

Die Beurteilung des jeweiligen Software-Entwicklungsprozesses sowie die Ableitung des entsprechenden CMM-Levels wird durch das „Software Engineering Institute“ oder einem von ihnen beauftragten Lizenznehmer durchgeführt. In der 5 Tage dauernden Hauptuntersuchung wird mit verschiedenen Methoden, wie z. B. Fragebögen und Interviews, die Erfüllung der Key Practices in den verschiedenen Key Process Areas festgestellt. Danach wird das Ergebnis dem Management präsentiert.

Selbst mit Hilfe dieser Instrumente bleibt es kompliziert und aufwendig, die Softwareentwicklung zu optimieren. Um den Softwareentwicklungsprozeß zu optimieren, also zu planen, zu steuern und zu überwachen, ist etliche Arbeit sowie der Einsatz von unter Umständen nicht unerheblichen Ressourcen erforderlich, aber nur so kann regelmäßig und zuverlässig Software hoher Qualität termingerecht im Rahmen der geplanten Kosten entstehen. Langfristig kann sich das Unternehmen dadurch besser im Markt etablieren und unzufriedene Kunden vermeiden. Die durch die Optimierung erreichte Qualitätssteigerung wirkt sich positiv auf die dem Kunden entstehenden Kosten aus.

## Einsatz

Das „Software Engineering Institute“ hat ein Modell zur Beurteilung und Optimierung dieser Prozesse geliefert, aber es liegt an dem verantwortlichen Management der Industrie, dieses Instrument effektiv in der Praxis zu verwenden. Die leider häufig schlechte Qualität von Softwareprodukten zeigt, daß insbesondere bei großen Projekten diese Aufgabe nicht vernachlässigt werden darf, da hier aufgrund der hohen Komplexität das Projektziel nicht mehr zuverlässig durch Einzelleistungen erreicht werden kann. Hier ist effektive, zielgerichtete Teamarbeit absolut notwendig.

## **Zusammenfassung des Capability Maturity Model**

Die folgende Tabelle liefert eine Kurzbeschreibung der Reifegrade, und deren Merkmale:

CMM-Level	Merkmale	Vorteile
5 (optimized)	<ul style="list-style-type: none"> <li>• kontinuierliche Prozeßverbesserung</li> <li>• Datensammlung und Analyse um Schwachpunkte zu finden und auszumerzen</li> <li>• Analyse und Verhütung von Problemen</li> </ul>	<ul style="list-style-type: none"> <li>• hohes Niveau der Prozeßpflege</li> <li>• hohe, gesicherte Produktivität</li> </ul>
4 (managed)	<ul style="list-style-type: none"> <li>• kontrollierte Prozesse</li> <li>• erste Qualitätsmetriken</li> <li>• umfangreiche Datenbasis mit Erfahrungen aus vorherigen Entwicklungen</li> </ul>	<ul style="list-style-type: none"> <li>• Nutzung technologischer Fortschritte</li> <li>• frühe Erkennung von Problemen und damit Möglichkeit deren Verhütung</li> </ul>
3 (defined)	<ul style="list-style-type: none"> <li>• definierte Prozesse</li> <li>• wiederverwendbare Module und Schnittstellen</li> </ul>	<ul style="list-style-type: none"> <li>• Qualitätsplan kann eingehalten werden</li> <li>• Geringes Risiko von Fehlschlägen</li> </ul>
2 (repeatable)	<ul style="list-style-type: none"> <li>• Abhängigkeit von Einzelpersonen</li> <li>• Minimale Prozeßkontrolle und Führung</li> <li>• hohes Risiko bei neuen Projekten</li> <li>• Mitarbeiterschulungen</li> <li>• Qualitätssicherung</li> <li>• Reviews und Tests</li> </ul>	<ul style="list-style-type: none"> <li>• nachvollziehbare Entwicklung</li> <li>• erste wirksame Führungsmechanismen</li> <li>• Abschätzung von Aufwand, Zeit und Kosten möglich</li> </ul>
1 (initial)	<ul style="list-style-type: none"> <li>• kein formalisierter Ablauf</li> <li>• Unverständnis bei den Entwicklern</li> <li>• Schwerpunkt liegt auf Produkt und nicht auf Entwicklung</li> <li>• Projektmanagement</li> </ul>	<ul style="list-style-type: none"> <li>• Management sieht Notwendigkeit zur Entwicklung der Entwicklung</li> <li>• Erkennbarkeit grundlegender Schwachpunkte</li> <li>• erster Einsatz von Software Engineering</li> </ul>

Etliche publizierte Untersuchungen belegen den Nutzen für den Mehraufwand. Die systematische Bewertung und Verbesserung der Entwicklungsprozesse steigern mittel- bis langfristig gesehen die Qualität und Produktivität deutlich. Durch den richtigen Einsatz dieser Methoden kann gutes Software Engineering in der Praxis etabliert und optimiert werden.

## ***Probleme und Grenzen***

Das Modell verlangt, daß zum Erreichen eines bestimmten Reifegrades alle Forderungen dieses und aller darunter liegenden erfüllt sein müssen. In konkreten Fällen bringt das nicht unbedingt Nutzen für den Prozeß. Es kann dabei dann ein unnützer Overhead entstehen. Einige wesentliche und wichtige Schlüsselprozesse werden von diesem Modell nicht oder unzureichend berücksichtigt, so fehlt zum Beispiel die Integration eines Risikomanagements und die Wiederverwendung wird nicht ausreichend beachtet. Ferner werden emotionale, menschliche und teambezogene Aspekte nicht ausreichend einbezogen; das Modell ist zu technologiellastig. Menschliche Faktoren und Soziale Beziehungen sind jedoch sehr wichtig und können die Arbeitsfähigkeit und Produktivität von Teams jeder Art nachhaltig positiv wie negativ beeinflussen.

Das Modell schreibt ferner Aktivitäten vor, ohne deren Vorteile aufzuzeigen, oder deren Notwendigkeit zu belegen oder messen zu können. Damit ist auch dieses Modell nicht allein ausreichend; die Fähigkeit und Flexibilität des Managements bleibt von essentieller Bedeutung.