Eine geführte Tour durch die Landschaft der Software-Prozesse und -Prozessverbesserung

Martin Glinz

In den letzten zehn Jahren wurde die fundamentale Bedeutung von Software-Prozessen und -Prozessverbesserung für die Software-Qualität mehr und mehr erkannt. Entsprechend grosser Aufwand wurde in die Entwicklung entsprechender Modelle und Normen investiert, sodass sich heute eine sehr vielfältige (und zunehmend unüberschaubare) Landschaft präsentiert. Dieser Artikel gibt eine Einführung in die Konzepte der Software-Prozesse und -Prozessverbesserung sowie einen Überblick über die einschlägigen Modelle und Normen. Wir beginnen mit einer Beschreibung der Prinzipien prozessorientierter Software-Entwicklung und diskutieren Chancen und Risiken. Dann geben wir einen Überblick über Modelle und Normen für Software-Prozesse. In einem zweiten Teil erläutern wir zunächst das Prinzip der Software-Prozessverbesserung und den grundlegenden Verbesserungszyklus. Dann charakterisieren wir die einschlägigen Modelle und Normen.

In the last ten years, software processes and software process improvement have been recognized as fundamental constituents for the development and maintenance of quality software. Much effort has been devoted to the development of models and standards for software processes, process maturity determination and process improvement. This article gives an introduction to the concepts of software processes and process improvement and presents an overview of models and standards in this field. First, we introduce the principles of process-oriented software development and discuss both advantages and risks. Then we give an overview of software process models and standards. In a second part, we present the concept of process improvement and illustrate the basic improvement cycle. Finally, we survey models and standards for software process improvement.

1 Prozesse und Prozessverbesserung

1.1 Einleitung

Herstellung, Pflege und Betrieb von Software sind Prozesse. Dabei verstehen wir unter *Prozess* "eine Folge von Schritten, die zur Erreichung eines gegebenen Zwecks ausgeführt wird" [IEEE 90], oder etwas präziser "den Ablauf eines Vorhabens mit der Beschreibung der Schritte, der beteiligten Personen, der für diesen Ablauf benötigten Informationen und der dabei entstehenden Informationen".

In der Diskussion der Frage, wie man qualitativ hochwertige Software erstellen kann, hat sich die Aufmerksamkeit in den letzten zehn Jahren verstärkt den Prozessen zugewandt. Man betrachtet also nicht das eigentliche Produkt, sondern den Prozess seiner Entwicklung sowie seines Betriebs und Unterhalts.

Martin Glinz ist Professor für Informatik an der Universität Zürich. Er arbeitet mit seiner Forschungsgruppe hauptsächlich auf dem Gebiet des Requirements Engineerings. Er beschäftigt sich ferner mit Software Engineering und Software-Qualitätsmanagement.

In der Praxis kann typisch zwischen ad-hoc Prozessen, welche spontan, individuell und ungeregelt ablaufen, und systematischen Prozessen, deren Ablauf geplant und gelenkt wird, unterschieden werden.

Systematische Prozesse zeichnen sich insbesondere dadurch aus, dass sie klar definiert sind und dass nach diesen Definitionen auch gearbeitet wird. Man erreicht damit eine Vereinheitlichung der Arbeitsweise und geringere Abhängigkeit der Projekte vom Können individueller Entwickler. Projekterfolge bleiben nicht zufällig, sondern werden wiederholbar; die Qualität wird lenkbar.

Mit der Fokussierung auf Prozesse eng verwoben ist das Konzept der *Prozessverbesserung*, bei dem es darum geht, die einmal definierten Prozesse nicht für immer in gleicher Weise anzuwenden, sondern sie fortlaufend zu analysieren, Schwachstellen und Verbesserungsmöglichkeiten zu identifizieren und entsprechende Anpassungen in den Prozessabläufen vorzunehmen.

1.2 Gefahren prozessorientierten Arbeitens

Allerdings ist letztlich die Qualität der *Produkte* entscheidend; die Prozesse sind nur Weg (und Wegweiser) zu diesem

Ziel. Fokussiert man (zu) stark auf die Prozesse, so läuft man Gefahr, den Weg für das Ziel zu halten.

Je mehr Aufwand in die präzise Definition und genaue Befolgung von Prozessen investiert wird, desto grösser werden drei weitere Gefahren, die des Realitätsverlustes, der Prozessbürokratie und der Erstarrung. Realitätsverlust tritt dann auf, wenn Aufgaben, für die es keinen Prozess gibt, nicht mehr wahrgenommen werden und Projekte unfähig werden, solche Aufgaben zu erkennen und zu bearbeiten. Prozessbürokratie äussert sich in sturem, unreflektiertem Befolgen der Prozesse, unter Abtötung von Flexibilität, Kreativität und Individualität der Kunden, was oft ebenso bürokratische und benutzerfeindliche Produkte zur Folge hat. Die Gefahr der Erstarrung droht besonders dann, wenn mit viel Aufwand und Mühe erfolgreiche Prozesse etabliert worden sind. Weil der Aufwand so gross war und weil der Erfolg da ist, wagt es niemand mehr, das einmal Erreichte durch Veränderungen zu destabilisieren.

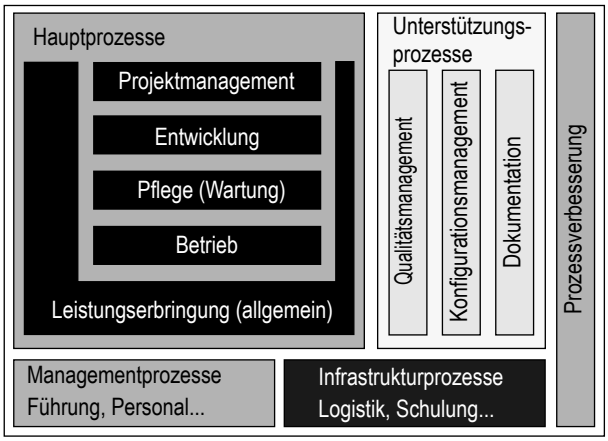

Abb. 1: Typische Prozessstruktur für ein Software-Unternehmen

1.3 Prozessdefinition

Prozessorientiertes Arbeiten setzt definierte Prozesse voraus und diese wiederum benötigen Prozessbeschreibungssprachen. Es gibt sehr unterschiedliche Möglichkeiten dies zu tun, je nachdem welche Eigenschaften von Prozessen man in den Vordergrund stellt.

- Sieht man Prozesse primär als Abläufe, so bietet sich eine ablauforientierte Darstellung an, zum Beispiel mit (an sich völlig veralteten, aber zum Teil immer noch populären) *Programmablaufplänen* bzw. einem dazu äquivalenten *Pseudocode* oder, etwas modernisiert und mit Sprachmitteln für parallele Abläufe erweitert, mit *UML-Aktivitätsdiagrammen* [Rumbaugh et al. 99].
- Stellt man den Aspekt der Koordination des ereignisgesteuerten zeitlichen Ablaufs paralleler und verteilter Aktivitäten in den Vordergrund, so bieten sich *Petrinetze* [Reisig 82] oder *Statecharts* [Harel 88] als Modellierungsmittel an.
- Betrachtet man Prozesse als das Fliessen von Information, insbesondere von Dokumenten durch ein Netz von informationsverarbeitenden Aktivitäten, so sind *Datenflussdiagramme* [McMenamin/Palmer 84] ein geeignetes Beschreibungsmittel für Prozesse.
- Eine weitere, in der Praxis allerdings selten benutzte Möglichkeit zur Prozessdefinition besteht darin, jeden Prozess durch eine Menge von *Regeln* zu beschreiben.
- In der Forschung sind ferner eine Reihe *formaler*, *ausführbarer* Prozessbeschreibungssprachen entwickelt worden (siehe zum Beispiel [Finkelstein et al. 94]), vor allem für den Einsatz in Workflowsystemen. (Workflowsysteme führen die Schritte eines Prozesses wie ein Programm aus und leiten beteiligte Personen bei der richtigen Prozessdurchführung an bzw. zwingen sie dazu.)

1.4 Prozess- und Vorgehensmodelle, Prozessnormen

Unternehmen, die ihre Software-Aktivitäten als Prozesse organisieren – und das dürfte mittlerweile eine Mehrheit sein – tun dies in der Regel etwa so wie in Abbildung 1 gezeigt.

Die einzelnen Prozesse werden unternehmensspezifisch oder teilweise auch projektspezifisch definiert oder sie folgen einem vordefinierten Vorgehensmodell. Ein bekanntes Beispiel für ein solches Vorgehensmodell ist das V-Modell [Bröhl/Dröschel 95], welches in Deutschland ein Standard für Behördenprojekte ist. Es hat seinen Namen von der V-förmigen Struktur des Entwicklungsprozesses (Abbildung 2), ist aber nicht auf den Entwicklungsprozess beschränkt, sondern regelt auch das Vorgehen im Projektmanagement, Qualitätsmanagement und Konfigurationsmanagement. Pendant zum V-Modell in der Schweiz ist HERMES [BFI 95, 96] der Standard für Behördenprojekte in der Schweiz.

Vorgehensmodelle legen die Prozesse in der Regel sehr detailliert fest, indem sie die durchzuführenden Aktivitäten, deren zeitliche Reihenfolge und die benötigten / zu bearbeitenden / zu erstellenden Dokumente vorschreiben. Das Ziel dabei

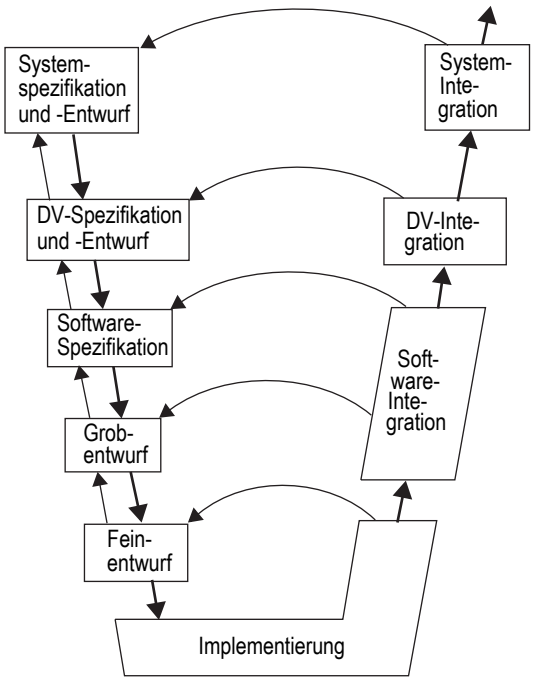

Abb 2: Der Software-Entwicklungsprozess im V-Modell

ist ein geregeltes, stabiles und wiederholbares Vorgehen, bei dem der Projekterfolg nicht mehr von besonderen Leistungen einzelner Individuen und von günstigen Umständen abhängt, sondern planbar und wiederholbar wird.

Gleichzeitig sind derart detailliert geregelte Prozesse aber auch der Hauptansatzpunkt für die Kritik an Vorgehensmodellen:

- Nicht jedes Projekt kann und soll nach dem gleichen Vorgehen abgewickelt werden. Vorgehensmodelle sind jedoch oft starr auf ein bestimmtes Prozessmodell fixiert. Das V-Modell beispielsweise hat bis vor kurzem strikt einen wasserfallartigen Entwicklungsprozess vorgeschrieben. Erst in der jüngsten Revision [Dröschel et al. 98] sind auch inkrementelle Prozesse zulässig.
- Vorgehensmodelle sind oft bürokratisch. Das tönt dann etwa so: "Sie müssen jetzt Vorgang A 2.35 bearbeiten. Hierzu benötigen Sie die Dokumente D1.14 bis D 1.17 sowie H1.6 und H1.9. Als Ergebnis müssen Sie die Dokumente D.211 und Kapitel 3.6 in D.26 erzeugen, sowie H1.9 revidieren." Da Software-Entwicklung zu wesentlichen Teilen eine kreative und interaktive Arbeit ist, ist ein solch starr geregelter Arbeitsfluss nicht adäquat.

Man kommt daher von festen Vorgehensmodellen wieder zunehmend ab und beschränkt sich in der Normierung auf die Festlegung eines Prozessrahmens, der unternehmens- und projektspezifisch zugeschnitten werden kann und soll. Von Bedeutung ist hier vor allem die Norm ISO/IEC 12207 ([ISO 95b], siehe Seite 13).

Die im Zusammenhang mit definierten und geregelt ablaufenden Prozessen häufig ins Spiel gebrachte Normenfamilie ISO 9000 ist in der gegenwärtig gültigen Fassung nicht prozessorientiert, wird aber häufig mit Prozessen eingesetzt (siehe Abschnitt 3.4 und Seite 14).

2 Software Prozessverbesserung – Das Konzept

Von Deming [Deming 86] stammt die Idee der kontinuierlichen Qualitätsverbesserung durch einen Zyklus, den er mit "Plan-Do-Check-Act" bezeichnet hat. Deming meint damit, dass Qualitätsmassnahmen zunächst geplant und dann nach Plan durchgeführt werden sollen. Während und nach der Durchführung wird der Erfolg der Massnahmen gemessen oder in anderer geeigneter Weise überprüft. Bei Abweichungen wird gehandelt. Dies können Massnahmen zur Korrektur der festgestellten Abweichungen, Planänderungen oder Verbesserungen im Qualitätsmanagementsystem sein.

Der Prozessverbesserung liegt ein analoges Konzept zu Grunde (Abbildung 3). Voraussetzung sind definierte Prozesse, nach denen auch tatsächlich gearbeitet wird. Alle Arbeiten zur Prozessverbesserung beginnen daher damit, dass die notwendigen Prozesse definiert werden und dass die Arbeit nach diesen Prozessdefinitionen geschult und institutionalisiert wird.

Ist das geschehen, so müssen die Prozesse gelebt und geführt werden. Prozesse führen heisst zunächst einmal, dass die Verantwortlichen die IST-Prozessabläufe systematisch beobachten und erheben. Dies geschieht einerseits kontinuierlich, indem wichtige Prozessgrössen gemessen oder mit anderen Mitteln erhoben und ausgewertet werden. Die Ergebnisse dienen primär zur Steuerung und Fortschrittskontrolle der laufenden Projekte. Durch Trendanalysen und Vergleiche über mehrere Projekte hinweg können aber auch Schwachstellen und Verbesserungspotenziale in den SOLL-Prozessen, d.h. in den Prozessdefinitionen erkannt werden.

Zusätzlich können die Prozessabläufe durch dafür geschulte Experten evaluiert werden. Dabei wird gezielt nach systembedingten Abweichungen zwischen IST- und SOLL-Prozessabläufen gesucht und es wird beurteilt, ob die SOLL-Prozesse für die damit zu lösenden Aufgaben angemessen sind. Auch hier ergeben sich Verbesserungspotenziale als Ergebnis.

Aus einer Analyse der so erkannten Schwachstellen und Verbesserungspotenziale ergeben sich nun konkrete Änderungsmassnahmen zur Verbesserung der Prozesse. Dabei verfolgt man typisch drei Ziele: Erstens sollen die Prozesse regelmässig an das sich verändernde Umfeld (Art der bearbeiteten Projekte, Bedürfnisse der Kunden, verfügbare Technologie,…) angepasst werden. Zweitens sollen erkannte, konkrete Schwächen in den bestehenden Prozessdefinitionen eliminiert werden. Drittens fragt man sich immer wieder, wo, wie und wie weit bestehende Prozessabläufe automatisiert werden können.

Betrachtet man ein eingespieltes Prozessverbesserungsprogramm über längere Zeit, so erkennt man typisch drei zyklisch aufeinanderfolgende Phasen. In der *Konsolidierungsphase* werden ungeregelt und chaotisch ablaufende Prozesse stabilisiert und unter Kontrolle gebracht. In der darauffolgenden *stetigen* Phase werden die bestehenden Prozesse kontinuierlich und graduell verbessert, ohne aber die Prozesse selbst in Frage zu stellen. In der *Sprungphase* dagegen wird versucht, durch grundlegende Veränderungen (radikales Überarbeiten oder gar Aufgeben bestehender Prozesse, Definition neuer Prozesse,…) einen Quantensprung in der Qualität der Prozesse zu erzielen. Auf die Sprungphase muss immer eine *Konsolidierungsphase* folgen, um die durch den Sprung entstandene Unordnung zu beseitigen und alle Prozesse wieder zu stabilisieren. Dann kann sich wieder eine Phase stetiger Verbesserung anschliessen, usw.

Dabei muss natürlich immer auch nach Kosten und Ertrag gefragt werden: Das Geld wird letztlich mit den Produkten

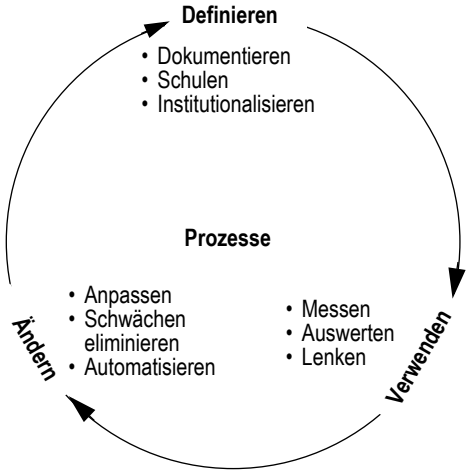

Abb. 3: Das Konzept der Prozessverbesserung

verdient, nicht mit den Prozessen. Untersuchungen zeigen jedoch, dass Prozessverbesserungsprogramme – trotz der damit verbundenen Kosten – zu teilweise erheblichen Kosteneinsparungen führen können [Dion 93], [Humphrey et al. 91].

3 Software Prozessverbesserung – Modelle und Normen

3.1 Grundlagen

Wer heute ein Prozessverbesserungsprogramm aufsetzen möchte, kann sich dabei auf eine Reihe vorhandener Modelle und Normen abstützen. Abbildung 4 gibt einen Überblick über die Prozessverbesserungslandschaft. Nachfolgend werden die einzelnen Modelle und Normen kurz charakterisiert. Für weiterführende Informationen wird insbesondere auf einschlägige WWW-Seiten [WWW 1], [WWW 2], [WWW 3] verwiesen. Auf die grundlegenden Software- und Qualitätsnormen, z.B. die Terminologienormen ISO 8402 und die IEEE 610.12 [ISO 95a], [IEEE 90], wird im Rahmen dieses Beitrags nicht eingegangen.

Jedes Software-Prozessverbesserungsprogramm benötigt *Masse*, einerseits um die Prozesse quantitativ erfassen und lenken zu können und andererseits, um die Reife der Prozesse und die angestrebte Verbesserung dieser Reife messen zu können.

GQM (Goal-Question-Metric) [Basili/Rombach 88], [Basili 92], [Fuggetta et al. 98] ist ein grundlegendes Verfahren zur zielorientierten Messung von Softwareprodukten oder -prozessen. Ausgehend von einem gesetzten Ziel werden Fragen formuliert, mit denen festgestellt werden kann, ob das gesetzte Ziel erreicht ist. Zu jeder Frage werden dann Masse bestimmt, welche eine quantitative Antwort auf die betreffende Frage erlauben.

ami (Assess, Analyse, Metricate, Improve) [Pulford et al. 95] war ursprünglich ein EU-ESPRIT-Projekt, welches zum Ziel hatte, systematische Verfahren zur Messung von Softwareprozessen praxistauglich zu machen und zu erproben. Jeder der vier Schritte von ami besteht aus drei Teilschritten. Im ersten Schritt (assess) werden die Prozesse beurteilt (z. B. mit CMM) und Verbesserungsziele definiert. Im Analyseschritt werden mit Hilfe von GQM entsprechende Masse hergeleitet. Im dritten Schritt (metricate) wird ein Messprogramm entwickelt und im vierten Schritt schliesslich werden die Messresultate ausgewertet und in Aktionspläne umgesetzt.

3.2 CMM und Umfeld

Das *Capability Maturity Model (CMM)* [Paulk et al. 93a], [Paulk et al. 93b], [Paulk et al. 93c] ist das älteste und wohl bekannteste Verfahren, das zur Verbesserung von Softwareprozessen eingesetzt wird. Es wurde Anfang der 90er Jahre vom Software Engineering Institute (SEI) geschaffen und basiert auf den Arbeiten von Humphrey [Humphrey 89]. Das dem CMM zugrundeliegende fünfstufige Bewertungsschema wurde ursprünglich dazu entwickelt, die Qualität der Softwareprozesse von Software-Lieferanten des US-Verteidigungsministeriums zu beurteilen. Das CMM wird zu einem Prozessverbesserungsprogramm, indem ein schrittweiser Aufstieg auf der CMM-Reifeskala geplant wird. Vom ursprünglichen, für die Beurteilung von Software geschaffenen CMM wurden diverse Varianten abgeleitet, u.a. ein Systems Engineering CMM, ein People CMM und ein Acquisition CMM.

In jüngster Zeit versucht das SEI, diese wieder unter einem gemeinsamen Dach, dem *CMMI*, zusammenzufassen. Näheres zu CMM siehe Seite 15 sowie [Stienen 99] auf den Seiten 16–22 in diesem Heft.

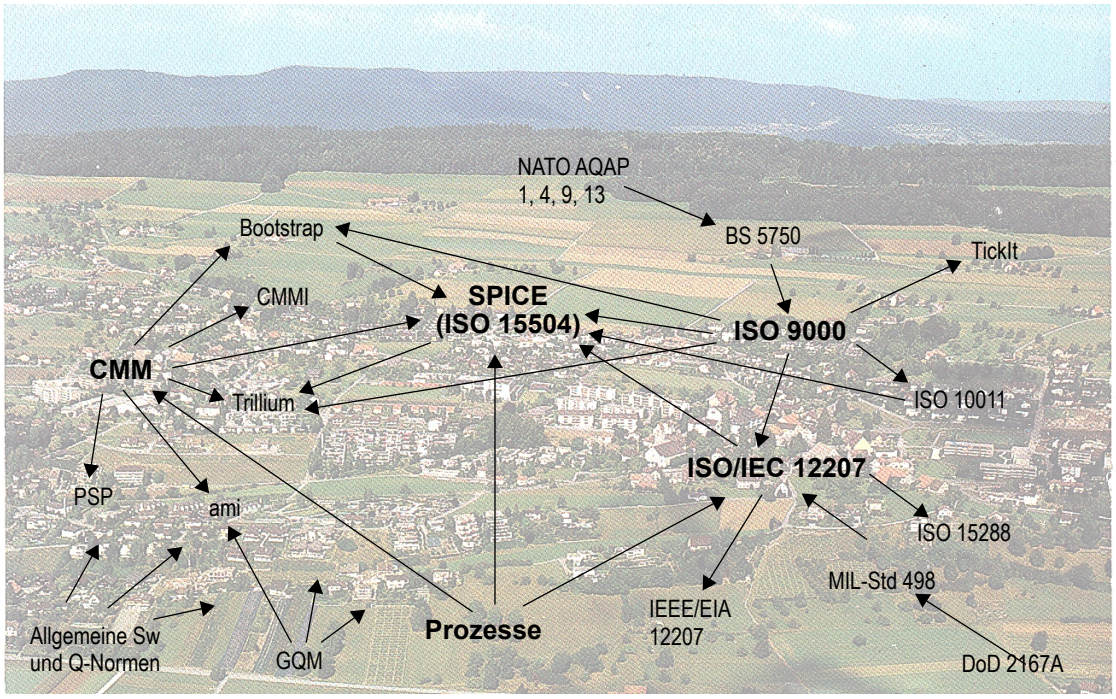

Abb 4: Die Software-Prozessverbesserungslandschaft

Bootstrap war ursprünglich ein EU-Projekt zur Prozessbeurteilung und -verbesserung [Haase et al. 94]. Bootstrap basiert auf dem CMM, nimmt Elemente von ISO 9000-3 hinzu und verwendet anstelle der fünf Reifestufen des CMM differenzierte Prozessreifeprofile zur Darstellung der Ergebnisse. Prozessbeurteilungen nach Bootstrap werden heute vom Bootstrap Institute vermarktet [WWW 4].

Trillium ist eine von Bell Canada entwickelte Variante des CMM, die auf die Anwendung in der Telekommunikationsindustrie zugeschnitten ist [WWW 8].

Der *Persönliche Softwareprozess* (PSP) [Humphrey 95] ist ein von Watts Humphrey, dem Vater des CMM, entwickeltes Prozessverbesserungsprogramm auf der Stufe individueller Entwicklerinnen und Entwickler. Während alle anderen Ansätze zur Prozessverbesserung auf der institutionellen Stufe ansetzen und die Prozesse des Unternehmens oder von Teilen davon betrachten, ist es das Ziel des PSP, die persönliche Produktivität und Qualität der einzelnen Entwicklerinnen und Entwickler zu verbessern.

3.3 SPICE und Umfeld

SPICE (Software Process Improvement and Capability dEtermination) ist ein 1992 gestartetes Programm mit dem Ziel, eine internationale Norm zur Prozessbeurteilung und –verbesserung zu entwickeln [WWW 6]. Parallel zur Entwicklung der Norm wurden umfangreiche Erprobungen durchgeführt. Die Norm trägt die Nummer ISO/IEC 15504 und liegt in Form von neun Technical Reports (einer Vorstufe zur endgültigen Norm) vor [ISO 98]. SPICE umfasst drei Elemente: Prozessbeurteilung (process assessment), Prozessverbesserung (process improvement) und Ermittlung des Prozesspotenzials (process capability determination). Die Prozesse werden in fünf Kategorien mit insgesamt etwa 40 Prozessen gegliedert. Zur Bestimmung der Prozessreife werden sechs Stufen mit neun gemeinsamen Attributen (generische Praktiken) verwendet. Die Reife jedes Prozesses einer Organisation wird einzeln bestimmt und zu einem Profil verdichtet. Mehr zu SPICE findet sich in "Nach CMM und Bootstrap: SPICE; Die neue Norm für Prozessbewertungen" [Stienen 99] in diesem Heft, Seite 16.

3.4 ISO 9000 und Umfeld

Die *ISO 9000* Normenfamilie ([ISO 1993, 1994a–c], siehe Seite 14) ist wie bereits oben erwähnt, nicht prozessorientiert, wird aber in der Regel prozessorientiet eingesetzt. Das heisst, Unternehmen, die ein Qualitätsmanagementsystem nach EN ISO 9001 einführen und zertifizieren lassen, entscheiden sich meistens für eine prozessorientierte Organisation ihrer Software-Aktivitäten. Wo bereits Prozesse existieren, ist eine ISO-9001-Zertifizierung oft Anlass für den Start eines Prozessverbesserungsprogramms. Dieser prozessorientierten Anwendung trägt die zur Zeit in Arbeit befindliche Totalrevision der ISO 9000 [WWW 5] Rechnung, indem diese prozessorientiert gestaltet wird.

Entstanden ist die ISO 9000 aus der britischen Norm BS 5750, die wiederum auf verschiedenen *NATO-Qualitätsnormen* (AQAP 1, 4, 9 und später 13) basiert.

TickIt ist ein britisches Programm zur Zertifizierung von Firmen nach ISO 9001 oder ISO 9002 unter Anwendung der ISO 9000-3 für alle softwarebezogenen Aktivitäten [WWW 7]. In Grossbritannien wird von Firmen, die Software entwickeln oder solche in ihre Produkte einbauen, ein TickIt-Zertifikat verlangt.

Im Zusammenhang mit ISO 9000 und SPICE / ISO 15504 ist ferner die Norm *ISO 10011*, eine Richtlinie zur Auditierung von Qualitätssystemen, von Bedeutung [ISO 90]. Audits sind ein wesentlicher Bestandteil jeder Implementierung der ISO 9000. Die Prozessbeurteilungsverfahren, welche in SPICE / ISO 15504 zur Anwendung kommen, sind verwandt mit der ISO 10011.

3.5 ISO 12207 und Umfeld

Die Norm ISO/IEC 12207 ([ISO 95b], siehe Seite 13) ist primär eine Software-Prozessnorm. Sie spielt eine wesentliche Rolle in der Software-Prozessverbesserung, indem jedes Software-Prozessverbesserungsprogramm ein Prozessmodell als Basis benötigt. SPICE / ISO 15504 ist beispielsweise ausdrücklich so aufgebaut, dass die ISO/IEC 12207 als Prozessmodell verwendet werden kann.

ISO/IEC 12207 hat zwei militärische Normen als Vorläufer. Die DoD 2167 A war von 1988 bis 1994 gültig und wurde dann durch MIL-Std 498 ersetzt. Dieser wiederum wurde 1998 ersetzt durch die IEEE/EIA 12207, die US-Implementierung der ISO/IEC 12207 [Ferguson/Sheard 98].

ISO 15288 ist eine geplante Norm mit dem Titel "Life-Cycle Management – System Life Cycle Processes". Sie soll auf der Ebene allgemeiner technischer Systeme das werden, was ISO/IEC 12207 für Software darstellt.

Schlussbemerkung

Hiermit sind wir am Ende unserer geführten Tour angelangt. Richtig angewendet, sind Software-Prozesse und -Prozessverbesserung ein zentrales Mittel für die systematische, disziplinierte Entwicklung und Pflege von Software. Die Fokussierung auf Prozesse darf aber nicht zu starrer, bürokratischer Arbeit nach Vorschrift entarten. Letztlich sind Prozesse und Prozessverbesserung nur Mittel zum eigentlichen Ziel, nämlich der Herstellung und Pflege guter Software.

Literatur

[Basili/Rombach 88]
V. Basili, D. Rombach (1988). The TAME Project – Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering* **14**, 6 (Jun. 1988). 758–773.

[Basili 92]
V. Basili (1992). *Software Modelling and Measurement: The Goal-Question-Metric Paradigm*. Technical Report, University of Maryland, CS-TR-2956.

[BFI 95]
Bundesamt für Informatik (Hrsg.) (1995). *HERMES – Führung und Abwicklung von Informatikprojekten*. EDMZ Art Nr. 609.201 d

[BFI 96]
Bundesamt für Informatik (Hrsg.) (1996). *HERMES SWE – Vorgehensmodell für die Software-Entwicklung*. EDMZ Art Nr. 609.211 d

[Bollinger/McGowan 91]
T.B. Bollinger, C. McGowan (1991). *A Critical Look at Software Capability Evaluations*. IEEE Software **8**, 4 (Jul. 1991). 25–41.
[Bröhl/Dröschel 95]
A. P. Bröhl, W. Dröschel (Hrsg.) (1995). *Das V-Modell: Der Standard für Softwareentwicklung mit Praxisleitfaden*. 2. Auflage. München: Oldenbourg.
[Deming 86]
W. E. Deming (1986). *Out of the Crisis*. Cambridge, Mass.: M.I.T. Press.
[Dion 93]
R. Dion (1993). Process Improvement and the Corporate Balance Sheet. *IEEE Software* **10**, 4 (Jul. 1993). 28–35.
[Dröschel et al. 98]
W. Dröschel, W. Heuser, R. Midderhoff (Hrsg.) (1998). *Inkrementelle und objektorientierte Vorgehensweisen mit dem V-Modell 97*. München, Wien: Oldenbourg.
[Ferguson/Sheard 98]
J. A. Ferguson, S. Sheard (1998). Leveraging your CMM Efforts for IEEE/EIA 12207. *IEEE Software* **15**, 5 (Sep/Oct. 1998). 23-28.
[Finkelstein et al. 94]
A. Finkelstein, J. Kramer, B. Nuseibeh (eds.) (1994). *Software Process Modelling and Technology.* New York: John Wiley & Sons.
[Fuggetta et al. 98]
A. Fuggetta, L. Lavazza, S. Morasca, S. Cinti, G. Oldano, E. Orazi (1998). Applying GQM in an Industrial Software Factory. *ACM Transactions on Software Engineering and Methodology* **7**, 4 (Oct. 1998). 411–448.
[Harel 88]
D. Harel (1988). On Visual Formalisms. *Communications of the ACM* **31**, 5 (May 1988). 514–530.
[Haase et al. 94]
V. Haase, R. Messnarz, G. Koch, H.J. Kugler, P. Decrinis (1994). Bootstrap: Fine-Tuning Process Assessment. *IEEE Software* **11**, 4 (Jul. 1994) 25–35.
[Humphrey 89]
W. S. Humphrey (1989). *Managing the Software Process*. Reading, Mass., etc.: Addison-Wesley.
[Humphrey et al. 91]
W. S. Humphrey, T. R. Snyder, R. R. Willis (1991). Software Process Improvement at Hughes Aircraft. *IEEE Software* **8**, 4 (Jul. 1991). 11–23.
[Humphrey 95]
W. S. Humphrey (1995). *A Discipline for Software Engineering*. Reading, Mass., etc.: Addison-Wesley.
[ISO 9000 98]
ISO (ed.) (1998). *Selection and use of ISO 9000*. International Standards Organization, Geneva. http://www.iso.ch/9000e/selusee.pdf
[ISO 9000 99]
ISO (ed.) (1999). *ISO 9000+ISO 14000 News*. Heft 3/99. http://www.iso.ch/9000e/revisions.pdf
[McMenamin/Palmer 84]
S. M. McMenamin, J. F. Palmer (1984). *Essential Systems Analysis*. New York: Yourdon Press.
[Paulk et al. 93a]
M. C. Paulk, B. Curtis, M. B. Chrissis, C. V. Weber (1993). Capability Maturity Model, Version 1.1. *IEEE Software* **10**, 4 (Jul. 1993). 18–27.
[Paulk et al. 93b]
M. C. Paulk, M. B. Chrissis, C. V. Weber (1993). *Capability Maturity Model for Software*. Version 1.1. Software Engineering Institute Technical Report CMU/SEI-93-TR-024, Feb. 1993.
[Paulk et al. 93c]
M. C. Paulk, C. V. Weber, S. M. Garcia, M. B. Chrissis, M. Bush (1993). *Key Practices of the Capability Maturity Model*, Version 1.1. Technical Report CMU/SEI-93-TR-025, Feb. 1993.
[Pulford et al. 95]
K. Pulford et al. (eds.) (1995). *A Quantitative Approach to Software Management: The ami Handbook*. Wokingham, etc.: Addison-Wesley.
[Reisig 82]
W. Reisig (1982). *Petrinetze - Eine Einführung*. Berlin-Heidelberg-New York-Tokyo: Springer.
[Rumbaugh et al. 99]
J. Rumbaugh, I. Jacobson, G. Booch (1999): *The Unified Modelling Language Reference Manual*. Reading, Mass., etc.: Addison-Wesley.
[Stienen 99]
H. Stienen (1999). Nach CMM und Bootstrap: SPICE: Die neue Norm für Prozessbewertungen. *Informatik/Informatique* **6**, 6.16–22

Normen

[IEEE 90]
IEEE (1990). Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990. IEEE Computer Society Press.
[ISO 90]
ISO 10011-1 (1990). Leitfaden für das Audit von Qualitätssicherungssystemen; Teil 1: Auditdurchführung.
[ISO 93]
ISO 9004-4 (1993). Qualitätsmanagement und Elemente eines Qualitätssicherungssystems; Teil 4: Leitfaden für Qualitätsverbesserung
[ISO 94a]
EN ISO 9000-1 (1994). Qualitätsmanagement- und Qualitätssicherungsnormen – Teil 1: Leitfaden zur Auswahl und Anwendung.
[ISO 94b]
EN ISO 9000-3 (1994). Qualitätsmanagement- und Qualitätssicherungsnormen – Teil 3: Leitfaden für die Anwendung von ISO 9001 auf die Entwicklung, Lieferung und Wartung von Software.
[ISO 94c]
EN ISO 9001(1994). Qualitätsmanagementsysteme: Modell zur Qualitätssicherung/QM-Darlegung in Design, Entwicklung, Produktion, Montage und Wartung.
[ISO 95a]
EN ISO 8402 (1995). Qualitätsmanagement – Begriffe.
[ISO 95b]
ISO/IEC 12207 (1995). Informationstechnik – Prozesse im Lebenszyklus von Software.
[ISO 98]
ISO/IEC TR 15504 (1998). Information technology – Software process assessment. Part 1–9.

WWW-Seiten

[WWW 1]
Verzeichnis von Verweisen zu Software-Prozessverbesserung und -Qualität: http://www.tantara.ab.ca/info.htm
[WWW 2]
Überblick über Modelle und Normen:
http://www.software.org/quagmire
[WWW 3]
Prozessnormen: http://www.cse.dcu.ie/essiscope/sm3/process/process.html
[WWW 4]
Bootstrap: http://www.bootstrap-institute.com/
[WWW 5]
ISO 9000:2000: http://www.bsi.org.uk/iso-tc176-sc2
[WWW 6]
SPICE: http://www.sqi.gu.edu.au/spice/
[WWW 7]
TickIt: http://www.tickit.org
[WWW 8]
Trillium: http://ricis.cl.uh.edu/process_maturity/trillium

ISO/IEC 12207

ISO/IEC 12207 [ISO 95b] gliedert die Softwareprozesse in drei Hauptgruppen (Abb. 1): Primärprozesse, Unterstützungsprozesse und Organisationsprozesse. Dabei handelt es sich nicht nur um die Prozesse für Software-Lieferanten. Der Beschaffungsprozess des Kunden sowie Prozesse, die vom Kunden und Lieferanten gemeinsam durchzuführen sind, gehören ebenfalls dazu. Auch der Betriebsprozess läuft sowohl beim Kunden wie beim Lieferanten ab. Abbildung 2 zeigt die Zusammenhänge zwischen den Prozessen. ISO/IEC 12207 ist ausdrücklich eine Rahmennorm, d.h. sie schreibt weder ein bestimmtes Prozessmodell noch die Art und den Aufbau der Dokumentation vor. Das heisst, dass sie für jedes Projekt individuell zugeschnitten werden muss, indem

- das Prozessmodell,
- die zu verwendenden Methoden, Sprachen und Werkzeuge sowie
- die zu erstellenden Dokumente

festgelegt werden.

Abbildung 3 zeigt das Verfahren des Zuschneidens (tailoring). Natürlich kann (und soll) ein Unternehmen für wiederkehrende Projekte der gleichen Art einen Standardzuschnitt definieren.

Abb. 1: Das ISO/IEC 12207 Prozessmodell

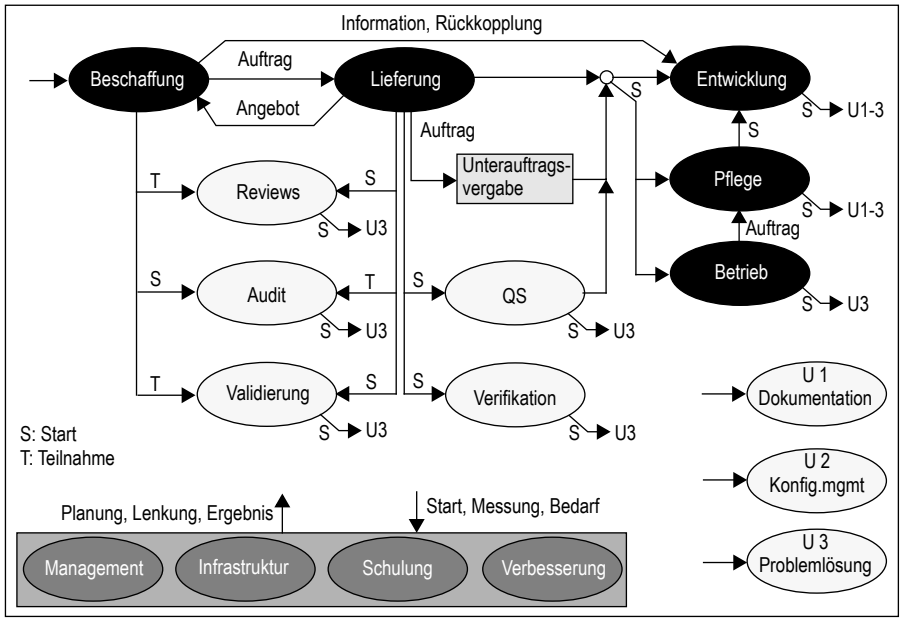

Abb. 2: Zusammenhang zwischen den Prozessen im ISO/IEC 12207 Prozessmodell

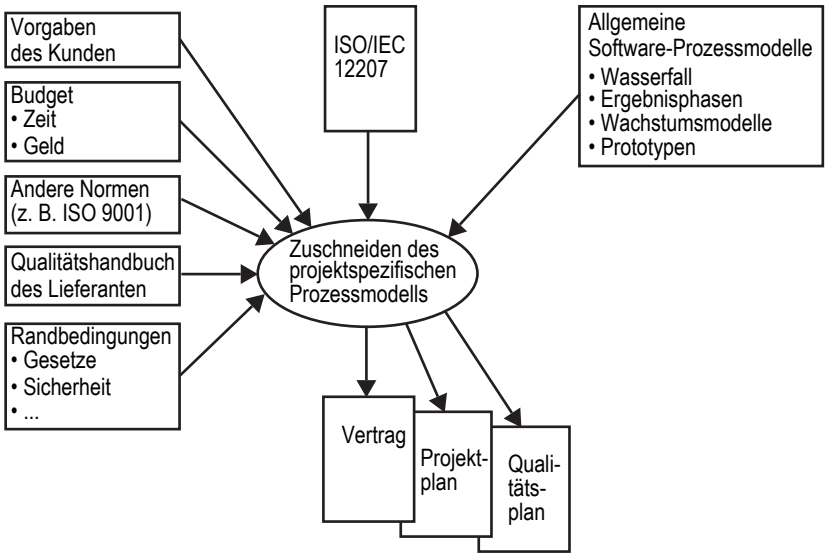

Abb. 3: Zuschneiden des ISO/IEC 12207 Prozessmodells auf ein Projekt

ISO 9000

Unter dem Namen ISO 9000 fasst man eine Sammlung von mittlerweile elf Qualitätsmanagement-Normen zusammen. Eine Übersicht über alle Normen der 9000-Familie gibt eine ISO Broschüre [ISO 9000 98]. Im Kontext von Softwareprozessen und Prozessverbesserung sind vor allem die Normen ISO 9001 und ISO 9000-3 relevant. Die ISO 9001 beschreibt in 20 Punkten Minimalforderungen an das Qualitätsmanagementsystem eines Unternehmens (Tabelle 1), und ISO 9000-3 legt diese (allgemein geltenden) Forderungen auf Software um. Für Prozessverbesserungsprogramme ist ausserdem die ISO 9004-4, eine Richtlinie für Qualitätsverbesserungsprogramme, interessant. Sie enthält konkrete Methoden und Techniken zur Planung, Messung und Auswertung von Qualitätsverbesserungsmassnahmen.

Die ISO 9000 wird zur Zeit vollständig überarbeitet und entschlackt. Die neue Familie ISO 9000:2000 wird aus nur noch drei Normen bestehen [WWW 5]:

- ISO 9000:2000 Qualitätsmanagementsysteme – Grundlagen und Begriffe
- ISO 9001:2000 Qualitätsmanagementsysteme – Forderungen
- ISO 9004:2000 Qualitätsmanagementsysteme – Leitfaden zur Qualitätsverbesserung

Die Veröffentlichung der neuen Normen ist für das vierte Quartal 2000 geplant. Eine gute Übersicht gibt [ISO 9000 99]; weitere Informationen hat es in [WWW 5].

Der wesentliche Unterschied zur bisherigen Normenfamilie ISO 9000 besteht in der Prozessorientierung (Abbildung 1). Insgesamt basiert die neue Normenfamilie auf acht Qualitätsmanagement-Prinzipien (Tabelle 2).

1.	Verantwortung der Leitung	11.	Prüfmittelüberwachung
2.	Qualitätsmanagementsystem	12.	Prüfstatus
3.	Vertragsprüfung	13.	Lenkung fehlerhafter Produkte
4.	Designlenkung	14.	Korrektur- und Vorbeugungsmassnahmen
5.	Lenkung der Dokumente und Daten	15.	Handhabung, Lagerung, Verpackung, Konservierung und Versand
6.	Beschaffung	16.	Lenkung von Qualitätsaufzeichnungen
7.	Lenkung der vom Kunden beigestellten Produkte	17.	Interne Qualitätsaudits
8.	Kennzeichnung und Rückverfolgbarkeit von Produkten	18.	Schulung
9.	Prozesslenkung	19.	Kundendienst
10.	Prüfungen	20.	Statistische Methoden

Tabelle 1: Die 20 Punkte der ISO 9001 (1994).

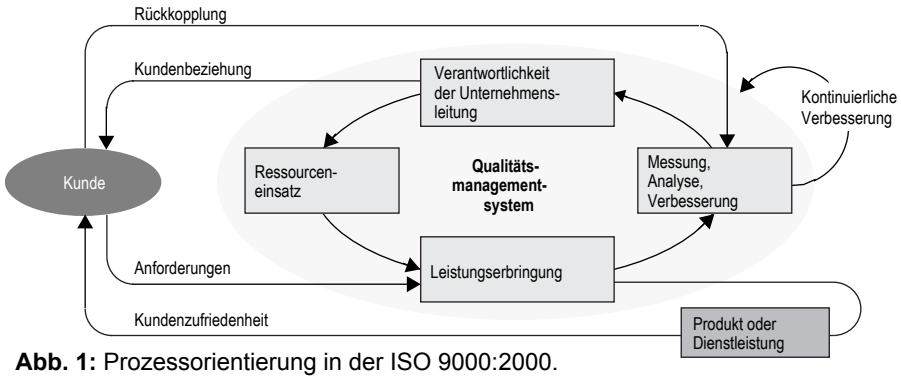

Abb. 1: Prozessorientierung in der ISO 9000:2000.

- Kundenorientierte Organisation
- Engagierte Führung (“Leadership”)
- Einbezug aller Beteiligten
- Prozessorientierung
- Systemischer Management-Ansatz
- Kontinuierliche Verbesserungen
- Faktenbasiertes Entscheiden
- Kunde-Lieferant-Partnerschaft

Tabelle 2: Die Qualitätsmanagement-Prinzipien der ISO 9000:2000.

Das Capability Maturity Model (CMM)

Grundlagen

Ursprünglich nur zur Bestimmung der Prozessreife von Software-Lieferanten geschaffen, ist das CMM heute eines der bekanntesten Modelle zur Beurteilung des Stands der Prozesse eines Software-Unternehmens und zur stufenweisen Verbesserung dieser Prozesse. Aktuell ist die Version 1.1 des Software-CMM [Paulk et al. 93a–c]; eine neue Version ist in Arbeit. Diese soll insbesondere die verschiedenen CMMs unter dem Dach des CMMI vereinigen (siehe Hauptartikel).

Das CMM basiert auf einer fünfstufigen Bewertungsskala (maturity levels) für die Prozessreife. Jede Stufe ist durch Schlüsselbereiche (key process areas) charakterisiert. Diese werden durch Schlüsselpraktiken (key practices) beschrieben.

Die Stufen des CMM

1 Ad-hoc (Initial)
- Ad-hoc-Prozess
- Keine formelle Planung und Kontrolle
- Projekterfolg ist abhängig von einzelnen Leuten mit herausragendem Können

2 Wiederholbar (Repeatable)
- Etabliertes Projektmanagement
- Abwicklung von Standardprojekten wird beherrscht; bei neuartigen Projekten grössere Risiken
- Keine etablierten Massnahmen zur Verbesserung des Prozesses

3 Definiert (Defined)
- Der Prozess ist klar definiert und läuft definitionsgemäss ab
- Es existiert eine Gruppe mit der Aufgabe, den Software Engineering Prozess zu lenken und zu verbessern

4 Geführt (Managed)
- Eine Mindestmenge an Qualitäts- und Produktivitätsmessgrössen wird erhoben und ausgewertet
- Es gibt eine Datenbank, in der alle relevanten Daten über den Prozess abgelegt werden und Mittel zur Pflege und Auswertung dieser Daten

5 Optimierend (Optimizing)
- Etablierter Regelkreis für Messung und Verbesserung des Prozesses
- Datenerhebung und Erkennung von Schwachstellen weitgehend automatisiert
- Durchgeführte Massnahmen aus dem erhobenen Datenmaterial begründet
- Etablierte Ursachenanalyse für alle Fehler und zugehörige Fehlerpräventionsmassnahmen

Bei der Beurteilung wird stufenweise vorgegangen: Zuerst werden nur die Kriterien der Stufe 2 geprüft. Wer hier durchfällt, ist auf Stufe 1. Wer besteht, wird zusätzlich nach den Kriterien der Stufe 3 beurteilt, usw.

Die Struktur des CMM

Abbildung 1 zeigt die Struktur des CMM zusammen mit einem Beispiel. Insbesondere die Gliederung der Reifegrade in Schlüsselbereiche und Schlüsselpraktiken wird dort deutlich.

Kritik am CMM

Die Kritik am CMM (siehe zum Beispiel [Bollinger/McGowan 91]) setzt vor allem bei folgenden Punkten an:
- Zuwenig differenziert: Die grosse Mehrheit der Software-Hersteller ist auf Stufe 1. Zwischen den schlechtensten und den besten auf dieser Stufe bestehen in Wahrheit grosse Unterschiede; diese werden vom CMM aber nicht erfasst.
- Zu anspruchsvoll: Es gibt kaum Unternehmen mit Software-Prozessen auf Stufe 4 und fast keine mit solchen auf Stufe 5.
- Zu einseitig: Das CMM orientiert sich am Idealbild von mittels statistischer Analyse von Prozessdaten gelenkten Software-Prozessen. Es ist jedoch keineswegs gesichert, ob dieses aus der Serienproduktion von Gütern entlehnte Prinzip generell für Software geeignet ist, oder ob es nur für Unternehmen sinnvoll ist, die sehr viele sehr ähnliche Projekte abwickeln.
- Stufenweise Beurteilung statistisch problematisch: Durch die stufenweise Beurteilung können wenige Schlüsselversager zur totalen Abwertung führen, auch wenn ein Unternehmen im Durchschnitt recht gut ist.
- Stufe 1 problematisch: Stufe 1 ist keine echte Stufe, sondern einfach der Ausschuss.

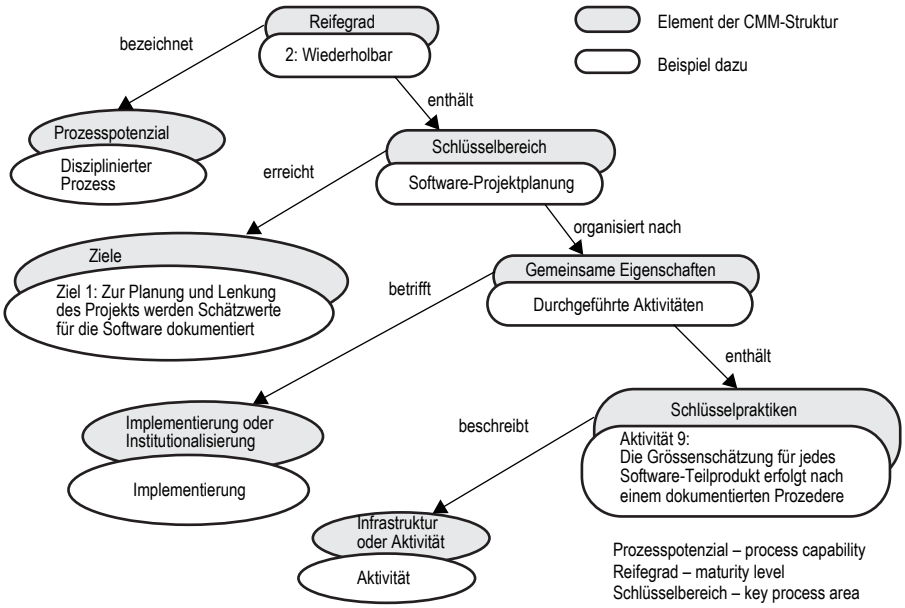

Abb. 1: Struktur des CMM