Seminararbeit

Der Persönliche und der Team Prozess (PSP/TSP)

Seminar Software Engineering

WS 03/04

Michael à Porta
99-700-155

Angefertigt am Institut für Informatik der Universität Zürich

Prof. Dr. M. Glinz

Betreuer: Ch. Seybold

INHALTSVERZEICHNIS

1. Einleitung

1.1 Probleme der Softwareentwicklung

Ein erfolgreiches Softwareprojekt zeichnet sich dadurch aus, dass es im geplanten Zeitrahmen, zu den definierten Kosten und in der erwünschten Qualität fertig gestellt wird. Die Realität zeigt jedoch, dass nur sehr wenige Softwareprojekte erfolgreich abgeschlossen werden (Stichwort: Softwarekrise). Eine umfassende Untersuchung der amerikanischen Standish-Group, die im so genannten „Chaos Report“ (vgl. [1]) veröffentlicht wurde, zeigt die wesentlichen Ursachen auf, weshalb Softwareprojekte scheitern.

Ursachen des Misserfolges (“Failure-Factors” Chaos-Report):

- Unvollständige, ungenaue und häufig sich ändernde Anforderungen
- Mangelnde Einbeziehung der Beteiligten
- Ressourcenmangel
- Unrealistische Anforderungen
- Mangelnde Unterstützung des Management
- Mangelhafte Planung
- Mangelndes IT Management
- Mangelndes Technologiewissen

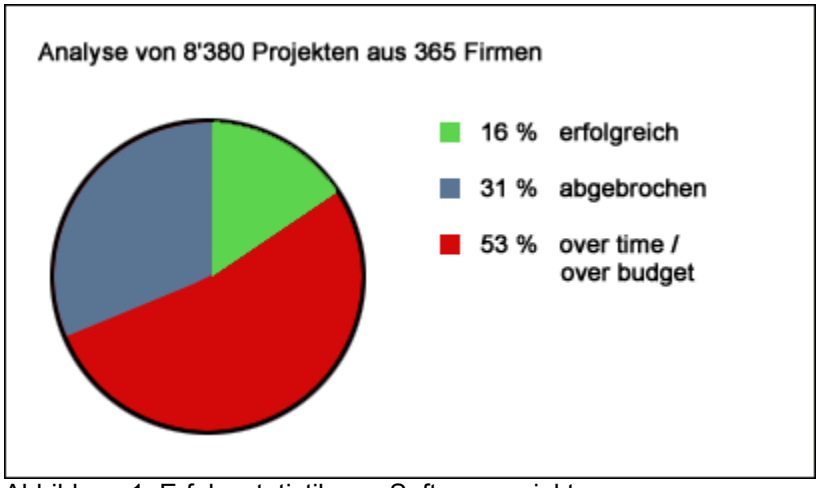

Abbildung 1: Erfolgsstatistik von Softwareprojekten

1.2 Hauptursache von Softwareproblemen

Die Vertreter des Software Engineering Institute (SEI) der Carnegie Mellon University sehen die Hauptursache des oben genannten Übels in der ungenügenden Qualität des Softwareprozesses und für den Schlüssel zu seiner Beseitigung.

Unter einem Prozess versteht man eine logisch zusammenhänge Folge von Aktivitäten zur Erstellung einer Leistung oder Veränderung eines Objektes.

1.3 Grundlage für einen guten Softwareprozess

Folgende Grundlage für einen guten Softwareprozess lassen sich identifizieren:

- **Wohldefiniertheit:**
  Dies ist die grundlegende Forderung an einen Softwareprozess und Grundvoraussetzung für einen geordneten Ablauf.

- **Quantitatives Verstehen:**
  Der Softwareprozess muss beobachtet und verstanden werden. Wo wird im Prozess wie viel Zeit verwendet und an welchen Stellen werden welche Fehler produziert.
  Dieses Verständnis bildet die Grundlage für die Prozessverbesserung.

- **Anpassungsfähigkeit**
  Der Softwareprozess muss in der Lage sein, sich kontinuierlich zu ändern. Einerseits um Verbesserungsmassnahmen vorzunehmen, aber auch um sich ändernden Anforderungen der Umgebung gerecht zu werden.

Auf diesen Erkenntnissen entwickelte das Software Engineering Institute (SEI) der Carnegie Mellon University Anfang der 90er Jahre im Auftrag des Pentagon das Capability Maturity Model.

1.3 Capability Maturity Model

Das Capability Maturity Model (CMM) dient der Bestimmung der Prozessqualität der Software-Entwicklung eines Softwareunternehmens. Es hilft Softwareunternehmen, ihre aktuelle Prozessreife zu ermitteln. Dadurch werden kritische Bereiche aufgezeigt, in denen Verbesserungen notwendig sind (vgl. [4]).

„Unreife“ Softwareunternehmung	„Reife“ Softwareunternehmung
• Unternehmen arbeitet „reaktiv“ • Zeitpläne und Budget werden überzogen • Keine formelle Planung und Kontrolle • Erfolge basieren oft auf Fähigkeiten eines einzelnen Ingenieurs	• Prozesse sind definiert • Arbeiten werden nach Plan durchgeführt • Zeitpläne und Budget basieren auf Referenzdaten • Kosten, Zeitpläne und Qualität werden üblicherweise eingehalten

Tabelle 1: Unterschied „Unreife“ / „Reife“ Softwareunternehmung

Das Capability Maturity Model gliedert den Weg von einer „Unreifen“ zu einer „Reifen“ Softwareunternehmung in fünf Reifestufen. Es ist auf organisationsweite Prozessverbesserung ausgelegt.

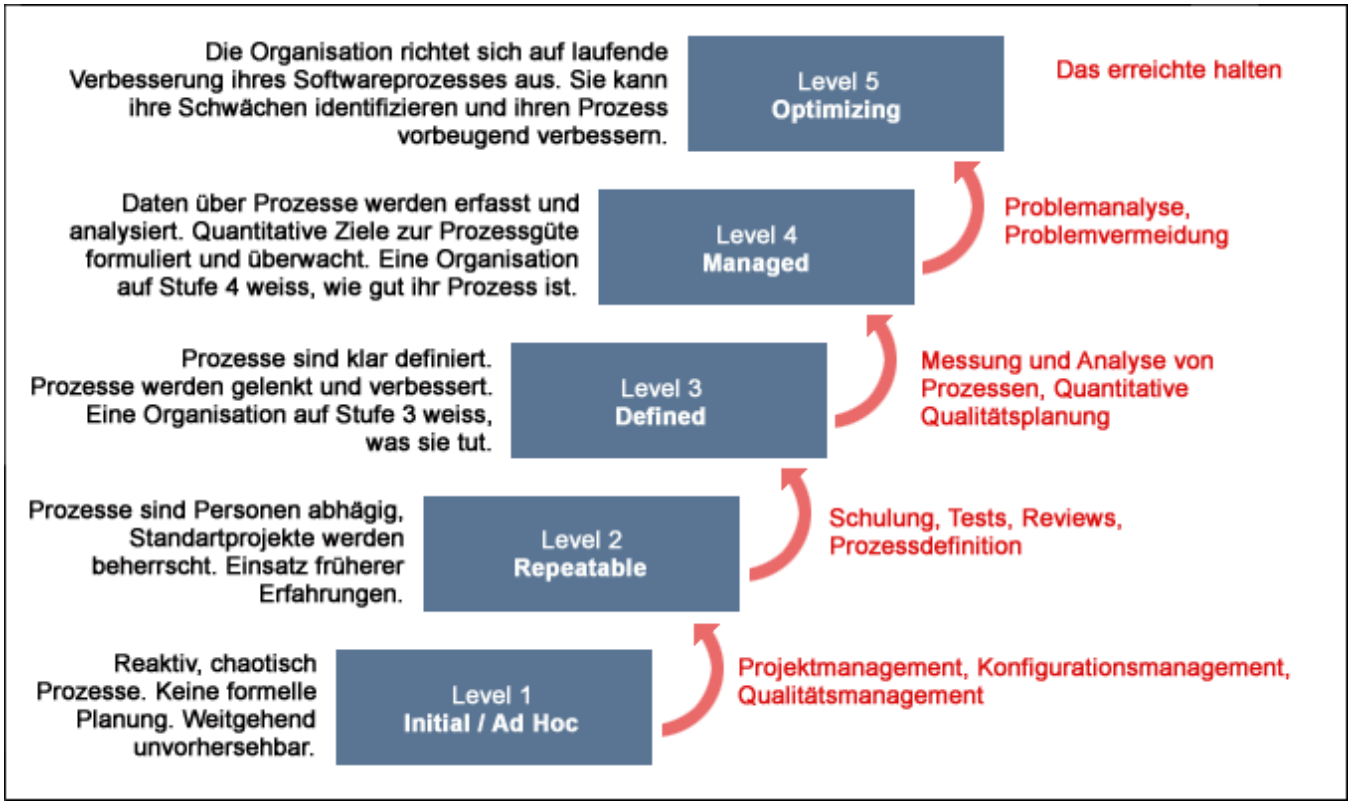

Abbildung 2: Capability Maturity Model

Das CMM liefert uns also ein organisationsweites Werkzeug zur Prozessverbesserung. Ein weiterer Aspekt zur Qualitätsverbesserung, sind die an den Prozessen beteiligten Personen, denn „Qualität beginnt mit den Menschen, nicht mit den Dingen“ (vgl. [3]). Zur Schliessung dieser Lücke wurde bereits 1995 von Watts S. Humphrey am Software

Engineering Institute (SEI) der Carnegie Mellon University aus dem CMM der Persönliche Softwareprozess (PSP) entwickelt. Der PSP ergänzt das organisationsweite CMM um eine individuelle Prozessverbesserung.

2. Der Persönliche Software Prozess

2.1 Einführung

Der Persönliche Software Prozess möchte einen Beitrag zur Prozessverbesserung auf der Ebene eines einzelnen Softwareingenieurs leisten. Dieses Ziel möchte er erreichen durch:

- Bessere Planung
- Präzises Feststellen der eigenen Leistung
- Verbesserung der Qualität der erstellten Produkte (geringere Defektzahlen im Produkt)

Zudem resultieren aus dem PSP brauchbare Informationen für das Management.

Wie der CMM ist auch der PSP in mehrere Stufen aufgeteilt, die nacheinander eingeführt werden. Sie führen den Entwickler von seinem aktuellen Softwareprozess zu einem selbst optimierenden persönlichen Softwareprozess. Die Stufen des PSP gehen von PSP0 bis PSP3. Bis zur letzten Stufe gehört zu jeder Stufe eine Zwischenstufe (vgl. [5]).

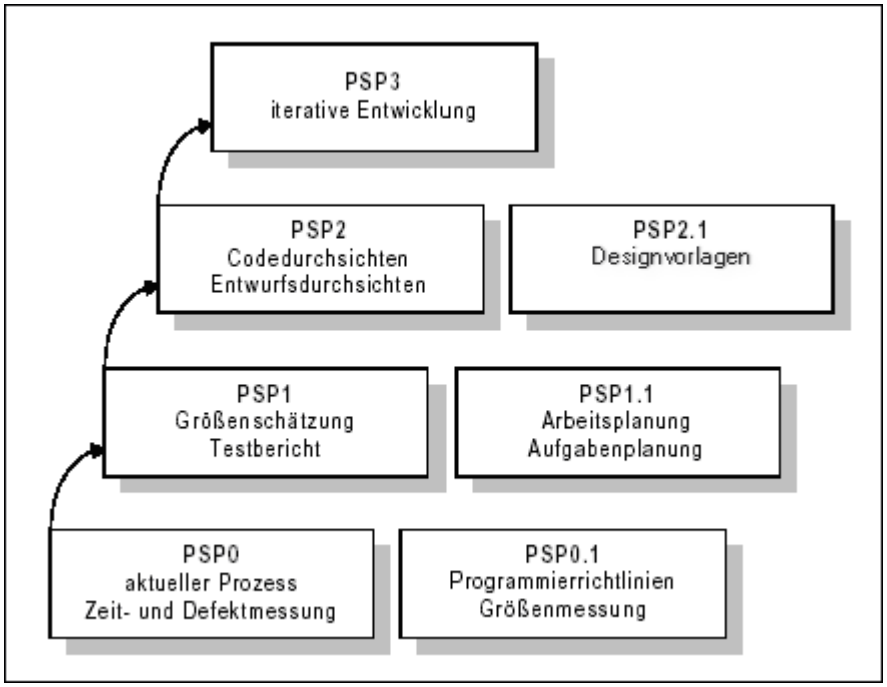

Abbildung 3: Struktur des persönlichen Software Prozess (PSP)

2.2 PSP Stufen

Der PSP ist geprägt durch eine Flut von Dokumenten, die dem Entwickler eine Anleitung zu seinem Handeln erteilen sollen. Er ist vergleichbar mit einem Kochrezept, in welchem alle Tätigkeiten penibel aufgelistet werden. Das Stufenmodell wurde eingeführt, um den Entwickler in Etappen an den zu erlernenden Prozess zu gewöhnen, es wird also auf jeder Stufe das bereits gelernte verfeinert und neue Elemente hinzugefügt. Im folgenden Abschnitt werden die Stufen des PSP besprochen und durch Grafiken auf die neuen Elemente im Entwicklungs-Prozess hingewiesen.

2.2.1 PSP0

PSP0 ist die Ausgangslage und geht von bereits verwendeten Prozessen des Entwicklers aus. Es wird angenommen, dass diese Prozesse bereits in Design, Programmierung, Übersetzung und Testen eingeteilt sind (vgl. Abbildung 4).

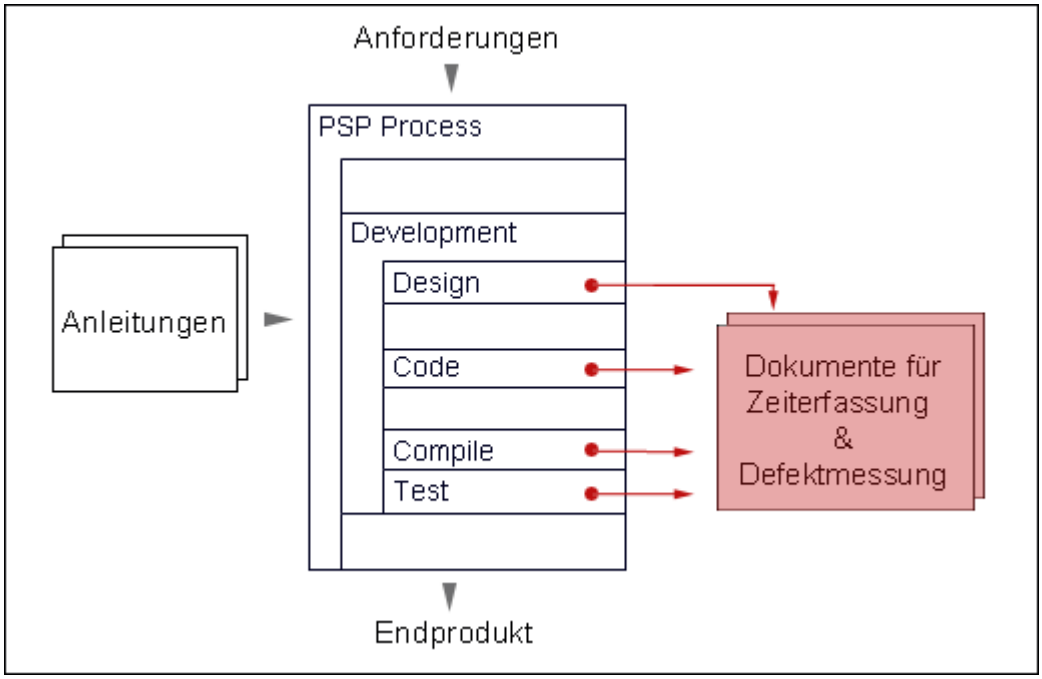

Abbildung 4: PSP0

Als erste Massnahme wird nun in PSP0 eine Zeiterfassung (vgl. Tabelle 2) und Defektmessung (vgl. Tabelle 3) eingeführt. Alle Messungen werden während der Arbeit vom Entwickler selbst vorgenommen.

Datum	Start	Ende	Unterbruch	Zeit total	Phase	Kommentar
5.11.03	09.00	10.30		90 min	Design	
	11.00	12.30	30 min	60 min	Code	3 Tassen Kaffee
	15.00	15.06	5 min	1 min	Compile	Telefon Mutter
	15.30	16.00		30 min	Test	

Tabelle 2: Zeiterfassung

Datum	Fehlertyp	Phase Defekt Ursprung	Phase Defekt Behebung	Zeit Für Behebung	Kommentar
5.11.03	20	Code	Compile	1 min	Fehlende Klammer im IF-Statement
	80	Code	Test	5 min	Resultat wird falsch berechnet
	70	Design	Test	15 min	Falsche Daten werden angezeigt

Tabelle 3: Defektmessung

Wert	Name	Beschreibung
10	Dokumentation	Kommentare, Ausgaben
20	Syntax	Schreibfehler, Tippfehler, Befehlsformate
30	Build, Paket	Änderungsmanagement, Bibliothek, Versionskontrolle
40	Zuweisung	Deklaration, doppelte Namen, Bereich, Grenzen
50	Schnittstelle	Prozeduraufrufe und -referenzen, IO
60	Überprüfungen	Fehlermeldungen, nicht ausreichende Überprüfungen
70	Daten	Struktur, Inhalt
80	Funktion	Logik, Zeiger, Schleifen, Rekursionen, Berechnung
90	System	Konfiguration, Timing, Speicher
100	Umgebung	Probleme mit Entwurfs-, Übersetzungssystemen

Tabelle 4: Defekttypen im PSP

Durch die Einführung von Zeiterfassung und Defektmessung können wir bereits folgende Merkmale berechnen:

- Benötigte Zeit für die Entwicklungsphasen
- Fehler pro Entwicklungsphase

2.2.2 PSP0.1

Auf der Stufe PSP0.1 werden zusätzlich folgende Ergänzungen eingeführt:

- Definition eines Quellcode-Standard (Grundlage für das Zählen der Quellcodezeilen)
- Grössenmessung (LOC)
- Vorschlagsformulare zur Prozessverbesserung, auf welchen die Entwickler Ideen zur Prozessverbesserung vermerken können

Dies liefert Informationen zur Fehlerrate und Entwicklungszeit (vgl. Abbildung 5).

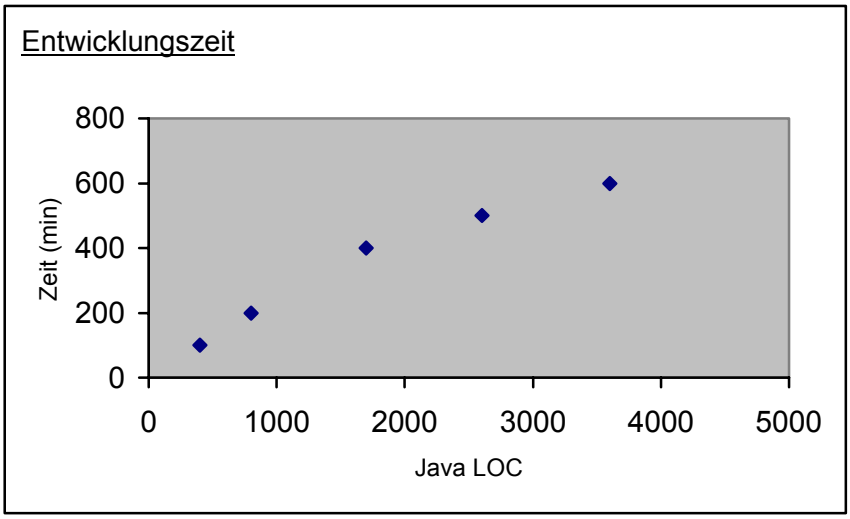

Abbildung 5: Entwicklungszeit

2.2.3 PSP1

In der PSP Stufe 1 wird der Prozess um eine Grössenschätzung (LOC) ergänzt. Je besser man in der Lage ist, den Aufwand und die Zeit für eine übernommene Aufgabe abzuschätzen, desto mehr kann sich ein Entwickler auf die pünktliche und korrekte Lösung der Aufgabe verlassen. Mit dieser Fähigkeit steigt also insgesamt die Leistungsfähigkeit.

Bekannte Verfahren zur Grössenschätzung sind:

- Delphi-Methode
- Function Points
- PROBE Methode (PROxy-Based-Estimating)
  Dieses Verfahren basiert auf einer Schätzung von Stellvertretern (z.B. Anzahl Methoden). Anhand dieser Stellvertreter wird unter Berücksichtigung historischer Daten eine Grössenschätzung vorgenommen.

Neben der Grössenschätzung wird auf dieser Stufe eine Formularvorlage für Testberichte eingeführt.

2.2.3 PSP 1.1

Diese Stufe ergänzt den PSP um die Arbeits- und Aufgabenplanung.
Aus den gesammelten historischen Daten ist es nun möglich, anhand der Grössenschätzung eine Zeitschätzung abzugeben. Der Entwickler erstellt einen Arbeitplan zusammen, aus welchem ersichtlich ist, wie viele Stunden er pro Woche an seinem persönlichen Projekt arbeiten kann. Dieser Arbeitsplan wird im Aufgabenplan bis auf die einzelnen Aufgaben verfeinert. Neben Anfangs- und Endtermin werden auch Meilensteine festgelegt.

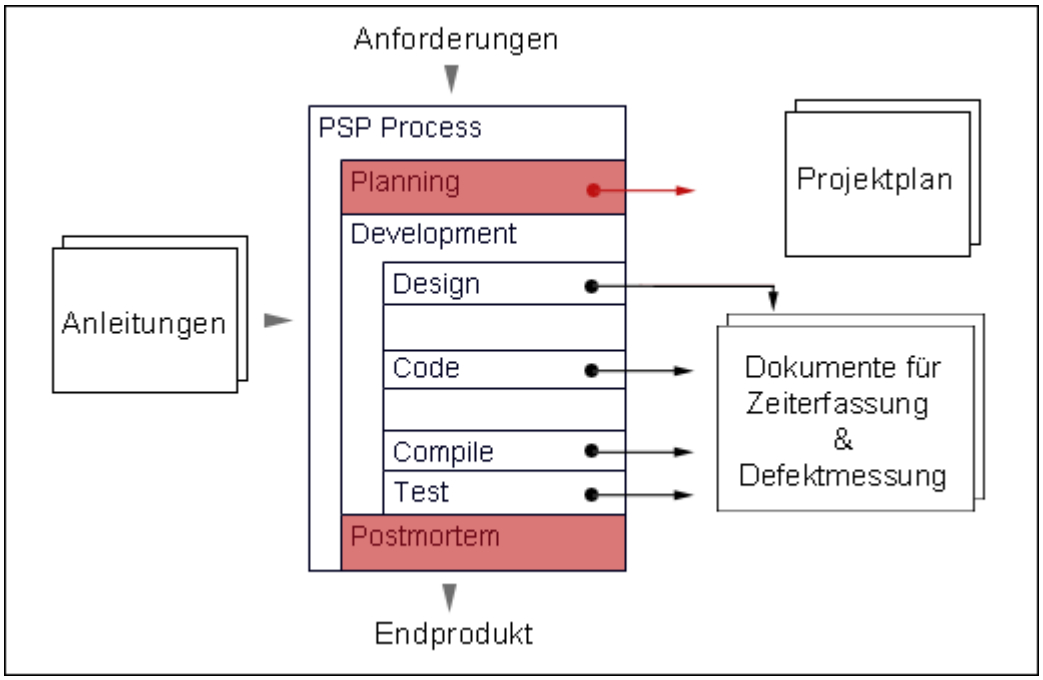

Abbildung 6: PSP1.1

Im Postmortem wird über das Projekt reflektiert. Die Messergebnisse aus der Ausführungsphase werden analysiert und als Projektzusammenfassung zu den historischen Daten hinzugefügt. Die Projekterfahrungen leben also auch nach dem „Tod” des Projekts weiter, daher der Begriff Postmortem.

2.2.5 PSP2

PSP2 ergänzt den PSP um Reviews auf Entwurfsebene und Codeebene, welche in Form der Selbstinspektion durchgeführt werden (vgl. Abbildung 7). Im Code Review kommt es zu einer Überprüfung von Syntax, Namensgebung, Speichermanagement und Funktionsaufrufen. Das Design Review beschäftigt sich mit Komplettheit, Programmlogik und Sonderfällen.
Es wird grosser Wert auf eine gründliche Durchführung gelegt, da Fehler möglichst früh zu finden sind, wenn sie noch mit geringem Aufwand behoben werden können.

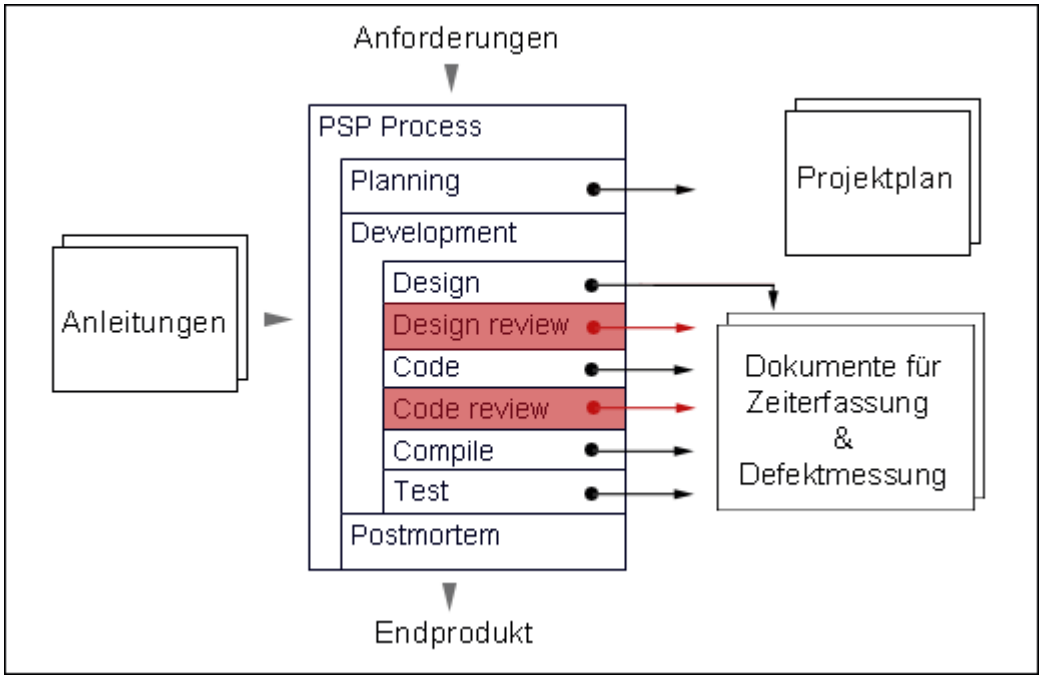

Abbildung 7: PSP2

Zur Selbstinspektion werden Checklisten verwendet, welche laufend durch gesammelte Fehlerdaten des Entwicklers angepasst werden, um die Aufmerksamkeit auf häufige Fehler zu lenken. Ziel ist es, so viele Fehler wie möglich vor dem ersten Test ausfindig zu machen, das heisst, getestet wird erst nach ausführlichen Inspektionen.

Eine Inspektion vor dem Testen durchzuführen, hat verschiedene Gründe:

- Wartezeit für Übersetzung sparen
- Qualität der Inspektion ermitteln
- Motivationsmittel für Entwickler

2.2.6 PSP2.1

Auf dieser Stufe werden Anforderungen an einen vollständigen Entwurf definiert.

Das Design wird im PSP in vier Schritten erstellt:

- Sammeln von Benutzeranforderungen
- Analyse der Anforderungen
- Erstellung des High-Level Designs
- Verfeinerung des Designs

Das High-Level Design entspricht einem Feinentwurf mit benötigten Datenstrukturen und detaillierter Funktionsweise der Methoden bis hin zu Pseudocode. Zudem sind Programmzustände und Interaktionen mit dem Benutzer beschrieben.

2.2.7 PSP3

Mit den oben besprochenen Stufen PSP0 – PSP2 lassen sich Programme bis 10'000 Zeilen bewältigen. Bei grösseren Programmen, sowie bei Entwicklungen im Team treten jedoch Probleme auf. Diese Stufe versucht das erste Problem zu lösen, indem das Gesamtsystem in kleinere Einheiten zerlegt wird.

3. Der Team Software Prozess

3.1 Einführung

Mit dem Persönlichen Softwareprozess haben wir eine Möglichkeit kennen gelernt, eine Prozessverbesserung auf der persönlichen Ebene zu erreichen. Zu Problemen kann es jedoch immer noch führen, wenn Entwickler im Team zusammen arbeiten müssen.

- Ineffiziente Führung
- Keine Kooperationsbereitschaft
- Mangelnde Kommunikation
- Fehlender Erfahrungsaustausch
- Schlechte Planung

Dieser Problematik soll der ebenfalls von Watts S. Humphrey entwickelte Team Software Prozess (TSP) Abhilfe schaffen (vgl. [5]). Eine der Hauptmotivationen für die Entwicklung des TSP war die Überzeugung, dass ein Team nur ausserordentliche Arbeit leisten kann, wenn

- es richtig gebildet wird,
- seine Mitglieder gut ausgebildet sind und
- es richtig geführt wird.

Diese drei Hauptelemente des TSP werden in Abbildung 8 verdeutlicht.

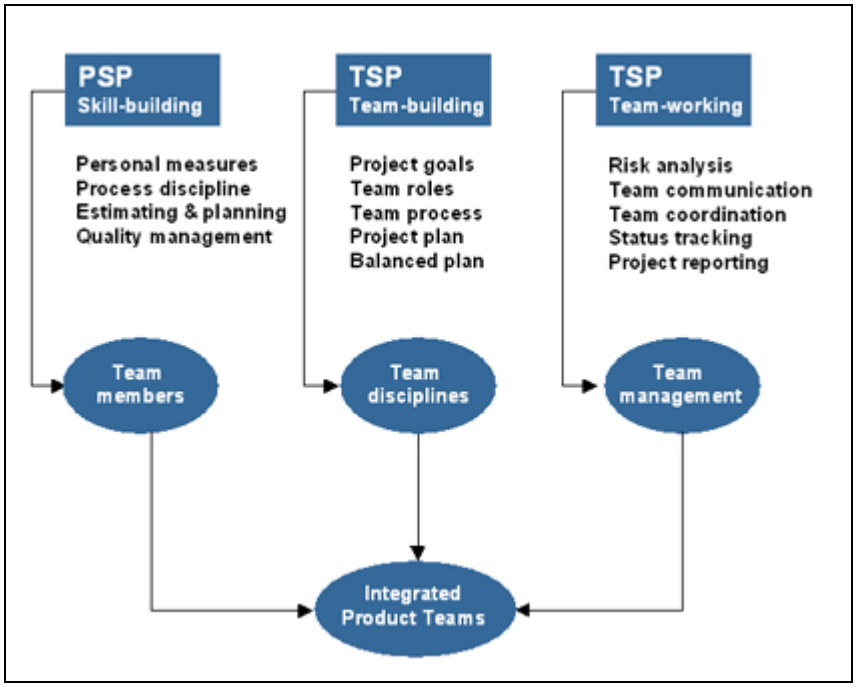

Abbildung 8: Hauptelemente des TSP

Bevor die Mitglieder an einem TSP teilnehmen können, müssen sie lernen, diszipliniert zu arbeiten. Sie müssen also den PSP beherrschen und in einem Training gelernt haben, wie man ausführliche Pläne erstellt und mit Prozessdaten umgeht.

Wenn diese Voraussetzung erfüllt ist, die Mitglieder also gut ausgebildet sind und ein einheitliches Prozessverständnis mitbringen, kann begonnen werden, ein Team zu errichten und zu führen, indem man folgende Regeln verwendet:

- Das Team stellt allgemeine Ziele auf und definierte die Rollen der Mitglieder.
- Das Team entwickelt eine Strategie.
- Das Team definiert einen allgemeinen Prozess für seine Arbeit.
- Alle Teammitglieder nehmen an der Planung teil.
- Jedes Mitglied kennt seine persönliche Rolle in diesem Plan.
- Das Team verhandelt über den Plan mit Management.
- Die Teammitglieder stehen frei und häufig in Verbindung.

3.2 Produkteinführung (Launch)

Um diese Regeln in die Realität umzusetzen, stellt der TSP eine ausdrückliche Anleitung von spezifizierten Schritten bereit (vgl. Tabelle 5). In einem viertätigen Planungsprozess, der Produkteinführung (Launch) genannt wird, entwickeln alle Teammitglieder gemeinsam die Strategie, den Prozess und den Plan ihres Projektes. Die folgende Tabelle soll dies verdeutlichen.

Tag	Schritt	Tätigkeiten	Beschreibung
1	1	Projekt- und Managementzielsetzungen	• Stellen Sie Teammitglieder vor • Besprechen Sie die Projektziele mit Management und stellen Sie Fragen
	2	Teamziele und -rollen	• Wählen Sie Teamrollen und Aushilfsrollen • Definieren Sie und dokumentieren Sie die Ziele des Teams
	3	Strategie entwickeln	• Definieren Sie die Entwicklungsstrategie • Definieren Sie den zu verwendenden Entwicklungsprozess
2	4	Gesamter Plan	• Entwickeln Sie Größenschätzungen und den gesamten Plan

	5	Qualitätsplan	• Entwickeln Sie den Qualitätsplan
	6	Ausgeglichener Plan	• Verteilung der Arbeit auf die Teammitgliedern • Bottom-Up Phasenpläne für jedes Teammitglied
3	7	Projekt Gefahren Analyse	• Kennzeichnen Sie und werten Sie Projektgefahren aus
	8	Report Vorbereitung	• Bereiten Sie einen Produkteinführungsreport für das Management vor
4	9	Management Bericht	• Wiederholen Sie Produkteinführungstätigkeiten und Projektpläne mit dem Management • Besprechen Sie Projektgefahren, Verantwortlichkeiten und geplante Tätigkeiten

Tabelle 5: Massnahmenkatalog für Produkteinführung (Launch)

Nach dieser Produkteinführung folgt das Team seinem eigenen definierten Prozess, um die Arbeit zu erledigen. Jedes Teammitglied besitzt nun einen ausführlichen Plan für die folgende Projektphase.

3.3 Die TSP Struktur

Da Ingenieure im Allgemeinen ausführliche Pläne nicht für mehr als ungefähr drei oder vier Monate bilden können, wird der Entwicklungsprozess in Zyklen/Einheiten aufgeteilt. Am Anfang eines jeden Zyklus findet dann erneut ein Relaunch statt, damit jeder Zyklus geplant werden kann auf dem Wissen, das im vorhergehenden Zyklus gewonnen wurde. Ein Relaunch sollte auch durchgeführt werden, wenn unerwartet Änderungen anstehen oder neue Mitglieder ins Team kommen.

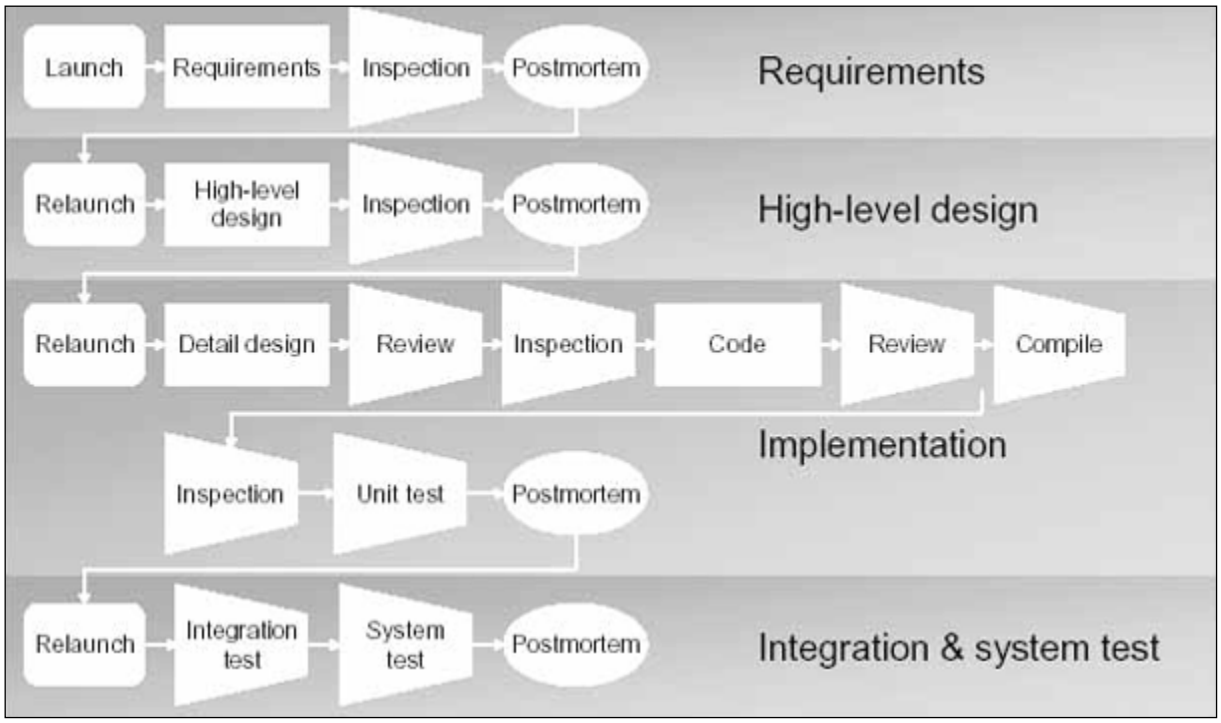

Abbildung 9: Der TSP Fluss als Wasserfallannäherung

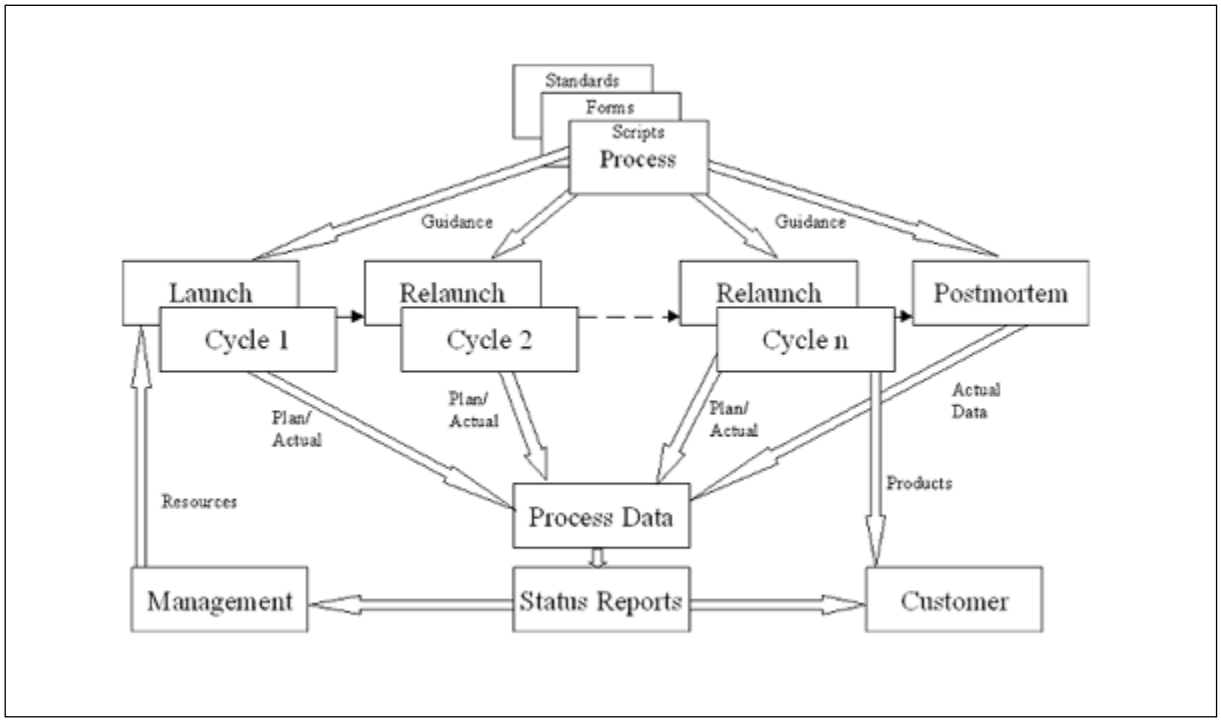

Abbildung 10: Der TSP Fluss mit zyklischer Prozessstruktur (Cycle 1 bis Cycle n sind in Phasen Strategie, Plan, Anforderungen, Design, Implementierung, Test und Postmortem eingeteilt.)

Wie im PSP findet auch im TSP eine laufende Messung der Prozesse statt. Zu diesem Zweck werden laufend Grösse, Zeit und Defekte gemessen. Diese Messungen haben denselben Zweck wie beim PSP. Man möchte den Teamprozess verbessern, indem man aus vergangenen Fehlern lernt und zudem bei Grössenmessungen auf historische Daten zugreifen kann.

Eine wichtige Rolle im TSP spielt der Teamleiter. Er ist für das Führen und Motivieren der Teammitglieder verantwortlich. Er stellt sicher, dass die Teammitglieder ihre Arbeit nach Plan erledigen, leitet wöchentliche Sitzungen, kommuniziert mit dem Management und dem Kunden und ist für eine offene und wirkungsvolle Teamkommunikation verantwortlich. Während der wöchentlichen Sitzung informiert er über den Projektstatus und Management-Themen. Teammitglieder berichten über ihren Arbeitsstand und was sie für die nächste Woche planen. Der TSP verlangt vom Team einen wöchentlichen Report, um aufzuzeigen, wo das Team hinter seinem Plan steht.

4. Erfahrungen aus der Praxis

4.1 Erfahrungen mit PSP

Die folgenden Abbildungen 11–13 sollen den Nutzen und Erfahrungen mit PSP verdeutlichen.

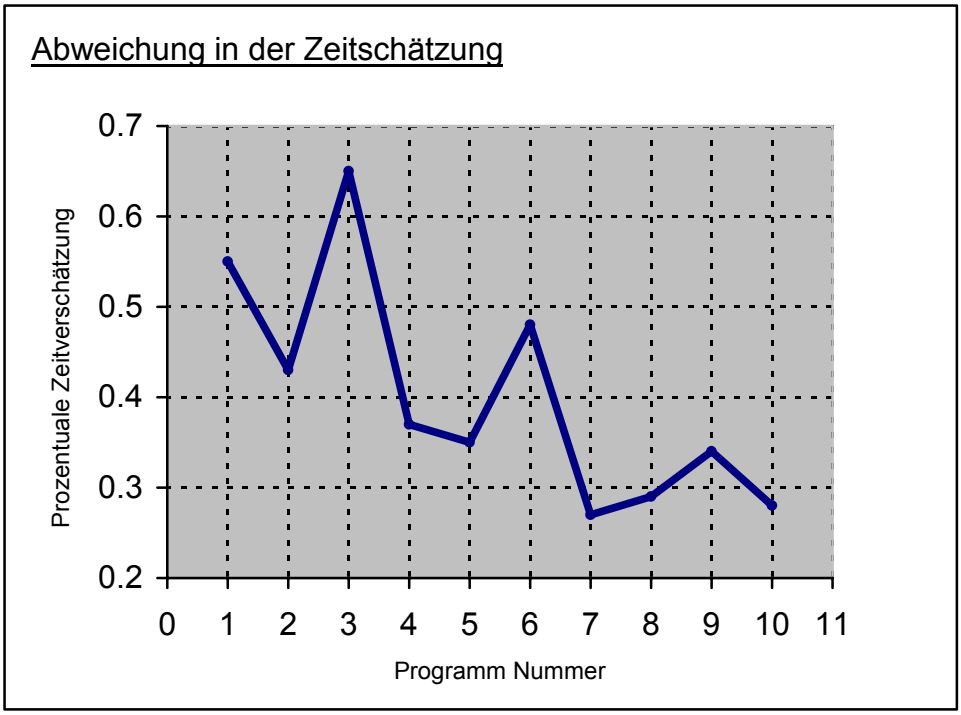

Abbildung 11. Abweichung in der Zeitschätzung

Untersuchungen haben ergeben, dass mit dem Erlernen des PSP die Entwickler in der Lage sind, genauere Schätzungen abzugeben. Dies wird in Abbildung 11 verdeutlicht. Ihre Abweichungen in der Grössenschätzung zu Beginn eines Lehrganges in PSP ist am linken Rand der Grafik zu erkennen, das Resultat am Ende der Schulung am rechten Rand. Diese Verbesserungen haben zur Folge, dass Entwickler bessere Zeitpläne für ihre Projekte erstellen können und dadurch ihren Zeitdruck reduzieren.

Wie in Abbildung 12 gezeigt, konnten die Defekte in den Phasen Compile und Testen drastisch von 110 Defekten pro 1’000 Linien Code (KLOC) auf 20 Defekte pro KLOC reduziert werden. Abbildung 13 zeigt, dass trotz grosser Verbesserungen in Planung und Qualitätsleistung die Linie der Codeproduktivität nicht abgenommen hat.

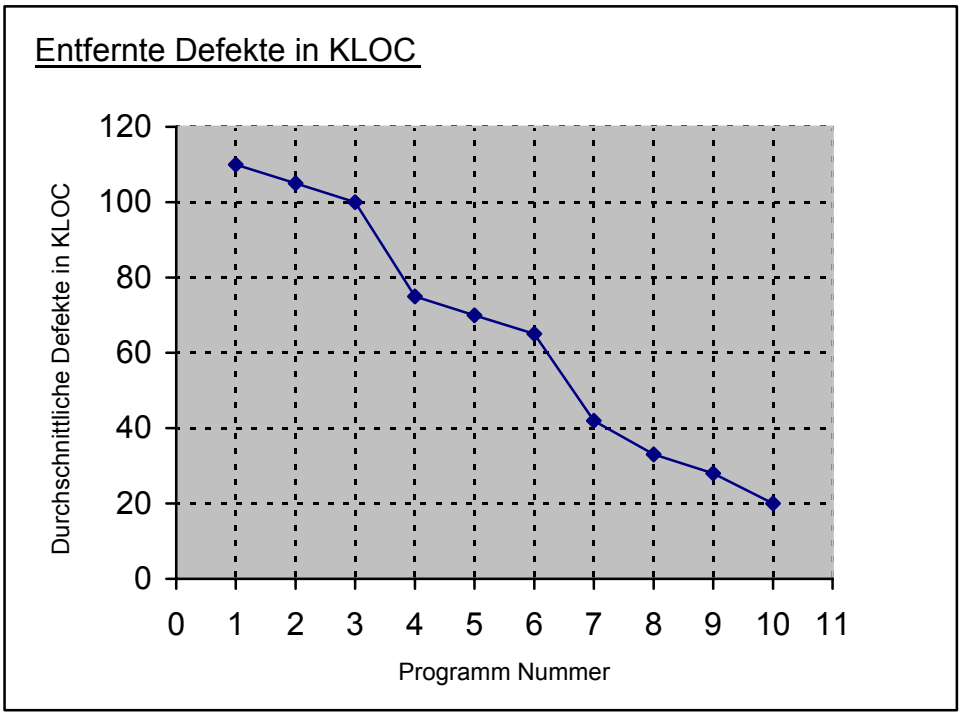

Abbildung 12: Qualitätsresultate

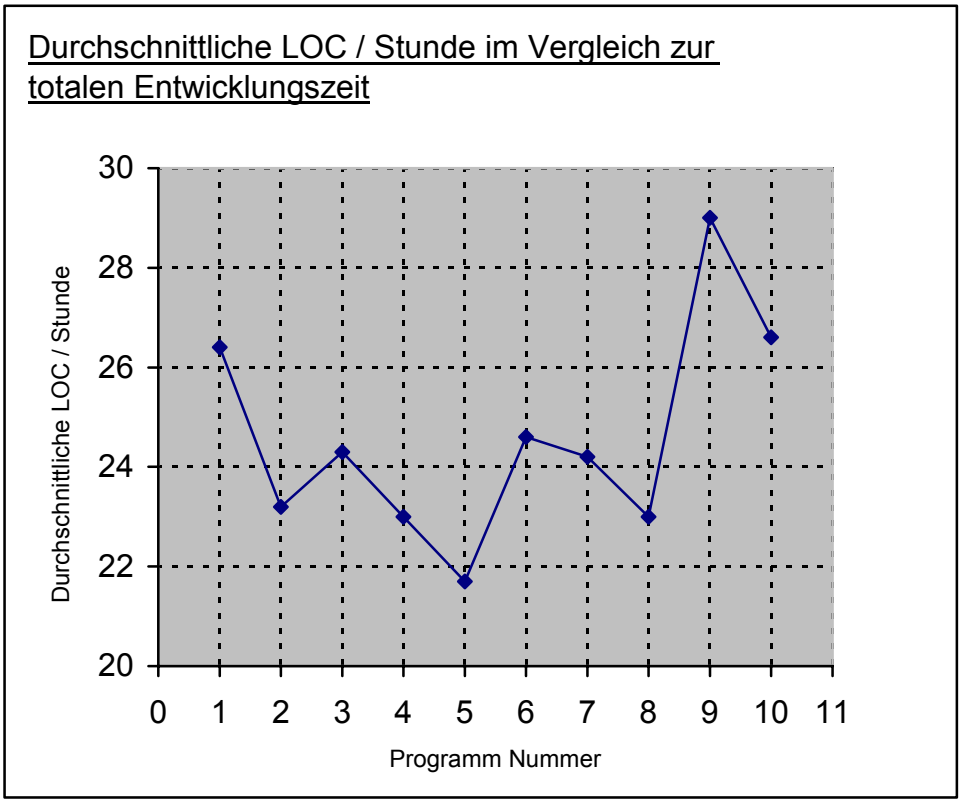

Abbildung 13: Produktivitätsresultate

4.2 Erfahrungen mit TSP

Obschon der TSP verhältnismäßig neu ist, kann bereits über vereinzelte Resultate durch eine Anzahl von Organisationen berichtet werden. Einige Beispiele werden in den folgenden Punkten zusammengefasst.

- **Teradyne,**
  der weltweit grösste Lieferant für automatische Testgeräte und führender Lieferant von Verbindungssystemen (vgl. [7]), hat herausgefunden, dass sie durch den Einsatz von TSP ihren durchschnittlichen Fehler-Level im Bereich Testen von 20 Fehlern pro KLOC auf 1 Fehler pro KLOC reduzieren konnten.

- **Hill Air Force Base,**
  in der Nähe von Salt Lake City, Utah, ist die erste Regierungsorganisation die CMM Stufe 5 erreichen konnte. Durch den erstmaligen Einsatz von TSP konnten sie die Teamproduktivität um 123% steigern und die Testzeit von einem Durchschnitt von 22% des Projektplans auf 2,7% reduzieren.

  .
- **Boeing**
  konnte in einem grossen Luftfahrtprojekt ihre Testphase um 94% reduzieren und stellte eine erhebliche Verbesserung im Projektzeitplan fest

5. Das Software Engineering Institute (SEI)

Das Software Engineering Institute (SEI) wurde 1984 gegründet und ist ein vom amerikanischen Verteidigungsministerium gesponsertes Forschungs- und Entwicklungszentrum. Das SEI ist an der Carnegie Mellon Universität in Pittsburg beheimatet.

Der Schwerpunkt liegt auf komplexen, verteilten, softwareintensiven, eingebetteten, echtzeitfähigen Systemen an die hohe Anforderungen bezüglich Zuverlässigkeit, Verfügbarkeit, Sicherheit, Leistung, Wartbarkeit und Kosten gestellt werden.

6. Watts S. Humphrey

Abbildung 14: Watts S. Humphrey

Watt S. Humphrey gründete das Software Prozessprogramm des Software Engineering Instituts (SEI) an der Universität Carnegie Mellon. Von 1959 bis 1986 arbeitete er bei IBM als Direktor der Abteilung Prozesse und Programmierqualität.

Seine Publikationen schliessen viele technische Referate und neun Bücher mit ein. Seine neuesten Bücher sind:

- Managing the Software Process (1989),
- Discipline for Software Engineering (1995),
- Managing Technical People (1996),
- Introduction to the Personal Software Process (1997),
- Introduction to the Team Software Process (2000),
- Winning with Software: An Executive Strategy (2001).

Herr Humphrey machte einen Bachelor Abschluss in Physik an der Universität von Chicaco, einen Master Abschluss in Physik am Institut für Technologie in Illinois und einen Master Titel in Business Administration an der Universität Chicaco.

1993 erhielt er vom amerikanischen Institut für Aeronautik und Astronauten den Aerospace Software Engineering Award. 2000 wurde das Watts Humphrey Software-Qualitätsinstitut in Chennai, Indien zu seinen Ehren benannt, und Boeing zeichnete ihn mit dem Preis für Führung und Innovation in der Verbesserung von Software Fertigungsprozessen aus.

7. Fazit

Abbildung 15 soll den Zusammenhang der besprochenen Themen nochmals verdeutlichen.

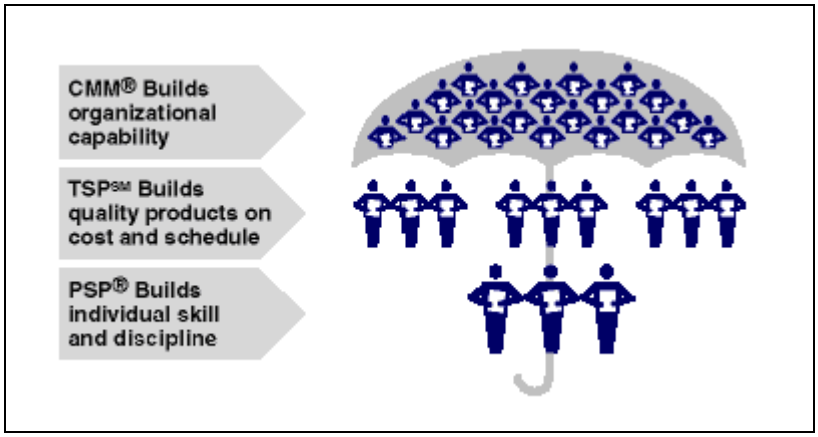

Abbildung 15: Zusammenhang

Wir haben besprochen, wie das CMM Organisationen helfen kann, ihre aktuelle Prozessreife zu ermitteln und dadurch die kritischen Bereiche aufzuzeigen, in denen Verbesserungen notwendig sind.
Ein wichtiger Aspekt ist jedoch, dass der Mensch im Mittelpunkt dieser Prozesse steht und nicht angenommen werden darf, dass er sein Handwerkszeug beherrscht. Die Zielsetzung des Personal Software Prozess ist daher, Entwicklern zu zeigen, wie sie Prozessgrundregeln in ihrer Arbeit verwenden. Da die meisten Softwareprodukte nicht durch eine einzige Person entstehen, muss der Entwickler lernen wie er nach vordefinierten Regeln und Abläufen mit anderen Entwicklern im Team zusammenarbeiten kann und Qualitätsprodukte unter konkurrenzfähigen Zeitplänen zu den geplanten Kosten produziert. Diese Grundlagen vermittelt der Team Software Prozess.

I Abbildungsverzeichnis

II Tabellenverzeichnis

III Literatur- und Quellverzeichnis

[1] The Standish Group International Inc.: The CHAOS Report (1994)
http://www.standishgroup.com/sample_research/chaos_1994_1.php
[2] Software Engineering Institute (SEI) der Carnegie Mellon University
http://www.sei.cmu.edu/cmm/
[3] P.B. Crosby, Quality is Free, McGraw-Hill, New York, 1979
[4] Software Engineering Institute (SEI) der Carnegie Mellon University
http://www.sei.cmu.edu/psp/
[5] Software Engineering Institute (SEI) der Carnegie Mellon University
http://www.sei.cmu.edu/tsp/
[6] Software Engineering Institute (SEI) der Carnegie Mellon University
http://www.sei.cmu.edu/publications/documents/doc.list/
[7] Teradyne
http://www.teradyne.com