

**Standards für die Bewertung**

**von Software-Entwicklungsprozessen**

**eingereicht von:**

Angelo Cavallaro

Beyza Gürler

David Süßlin

Hüseyin Kasarca

Nahom Ghebremussie

Viktor Sellmann

**eingereicht am:**

12. Februar 2020

**Begutachter:**

Herr Prof. Dr. German Nemirovskij

Herr Knut Kliem

Inhaltsverzeichnis

[Abstract 3](#_Toc32413613)

[1 Aktuelle Vorgehensansätze in der Praxis 3](#_Toc32413614)

[1.1 RUP (Rational Unified Process) 3](#_Toc32413615)

[1.2 V-Modell XT 4](#_Toc32413616)

[1.3 Wasserfall-Modell 6](#_Toc32413617)

[1.4 Agiles Vorgehen – SCRUM 6](#_Toc32413618)

[1.4.1 SCRUM-Allgemein 6](#_Toc32413619)

[1.4.2 SCRUM-Rollen 6](#_Toc32413620)

[1.4.3 SCRUM-Ablauf 7](#_Toc32413621)

[2 Zu beurteilende Softwarekriterien 8](#_Toc32413622)

[2.1 Softwarekosten 8](#_Toc32413623)

[2.2 Technische Anforderungen der Software 8](#_Toc32413624)

[2.3 Unternehmensspezifische Anforderungen der Software 9](#_Toc32413625)

[2.4 Benutzeroberfläche 9](#_Toc32413626)

[2.5 Softwaredokumentation 9](#_Toc32413627)

[2.6 Datensicherung 9](#_Toc32413628)

[2.7 Datenschutz 10](#_Toc32413629)

[2.8 Schnittstellen 10](#_Toc32413630)

[2.9 Performance 10](#_Toc32413631)

[3 Prozessstandards ISO 12207 und ISO 15288 10](#_Toc32413632)

[3.1 Softwareentwicklungsprozess 10](#_Toc32413633)

[3.2 Systemlebenszyklus ISO 15288 11](#_Toc32413634)

[3.3 Softwarelebenszyklus ISO 12207 11](#_Toc32413635)

[4 Softwareprozessverbesserungs- und Bewertungsmodelle – CMMI/SCAMPI, ISO 15504/ISO IEC 330xx-Normenreihe 15](#_Toc32413636)

[4.1 CMMI (Capability Maturity Model Integration) 15](#_Toc32413637)

[4.2 ISO 15504 (SPICE) & ISO IEC 3300x-Normenreihe 17](#_Toc32413638)

[4.3 Die Bewertungsmethode SCAMPI 20](#_Toc32413639)

[5 Anwendung am Fallbeispiel – Projektstudium SAPlexa 21](#_Toc32413640)

[5.1 Projektbeschreibung- & Ziel 21](#_Toc32413641)

[5.2 Vorgehensweise 21](#_Toc32413642)

[5.3 Kritische Betrachtung 21](#_Toc32413643)

[6 Fazit 22](#_Toc32413644)

[Quellen- und Literaturverzeichnis 23](#_Toc32413645)

# Abbildungsverzeichnis

# Abstract

Die Welt ohne Software ist in der heutigen Zeit unabdingbar. Für jedes noch so kleines Problem gibt es eine Software die das Problem löst. Doch wie so eine Software entsteht und welchen Arbeitsaufwand dahinter steckt wird kaum betrachtet. Die Software Entwicklung ist ein Prozess mit vielen Hindernissen, Fehler und auch Konflikte. Jedoch sollte stets die Effizienz im Vordergrund stehen, besonders in kostspieligen Projekten. Qualitätsmanagement befasst sich mit organisatorischen Maßnahmen, um eine hohe Produktqualität zu erreichen und nachzuweisen. Aus diesem Grund haben verschiedene Institutionen Normen auf diesem Gebiet entwickelt, die Unternehmen unterstützen, geeignete Ansätze und Verfahren auszuwählen sowie nach außen Vertrauen zu schaffen, um gewisse Mindestanforderungen zu erfüllen.

**Zielsetzung**

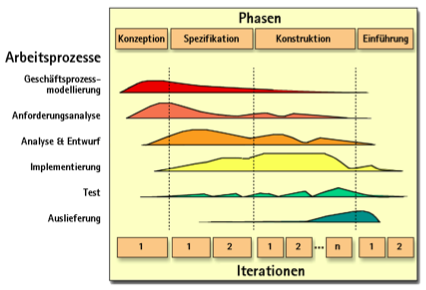
Das Ziel dieser Ausarbeitung ist es die Effizienz der Software Entwicklungsprozesse zu untersuchen, analysieren und zu bewerten. Dazu betrachten wir unser Fallbeispiel aus unserem Projektstudium „SAPLEXA“, mit der Problemstellung wie dieser Prozess kritisch analysiert zu bewerten ist.

### Vorgehensweise

Um dies kritisch beurteilen zu können wurden die aktuellsten Normen in diesem Gebiet zur Quelle gezogen und unterschiedlichste Facharbeiten zum Thema Qualitätsmanagement eingelesen.

# Aktuelle Vorgehensansätze in der Praxis

## RUP (Rational Unified Process)

****

Bei RUP werden die Arbeitsprozesse im Laufe des Entwicklungsprozesses in den Projektphasen Konzeption, Spezifikation, Konstruktion und Einführung iterativ mit unterschiedlicher Intensität und Dauer mehrmals durchlaufen.

Konzeption:

Diese Phase dient zum Ausformulieren einer Vision, eines Zieles und auch der Erstellung von Anwendungsfallmodellen, die die wesentlichen Funktionalitäten beschreibt.

Spezifikation

Hier wird der Architekturprototyp, genauso wie eine detaillierte Beschreibung der Anwendungsfälle ausgearbeitet. Außerdem erfolgt die Planung der Konstruktionsphase.

Konstruktion:

Nach der Ausarbeitung der Architektur konzentriert sich diese Phase vor allem auf die Entwicklung und das Testen des ausgearbeiteten Produkts. Hierbei entsteht die erste lauffähige Version des Produktes oder der Software.

Einführung:

Schlussendlich folgt nur noch die Übergabephase und Auslieferung des fertigen Produktes an den Kunden.

Quelle RUP David:

<https://deacademic.com/dic.nsf/dewiki/1161101>

Quelle ISO 15288 (und 12207)

<https://files.ifi.uzh.ch/rerg/arvo/ftp/papers/INF_6_99.pdf>

Quellen Beyza:

Wasserfallmodell Bild: <http://spolwig.de/is/softwareprojekte/slc_3.htm>

RUP modell Bild: <http://www.infforum.de/themen/anwendungsentwicklung/se-rup.htm>

Buch Grundkurs Wirtschaftsinformatik dietmar abs

## V-Modell XT

Das V-Modell XT ist ein Vorgehensmodell zur Organisation und Durchführung von Software- und Systementwicklungsprojekten und löste seine beiden Vorgänger erfolgreich ab. Den Anfang machte das im Auftrag des Bundes konzipierte V-Modell[[1]](#footnote-1), das den typischen Entwicklungsprozess in definierte Phasen gliedert. Dadurch ermöglicht man ein methodisches, phasenorientiertes Vorgehen mit integrierter Qualitätssicherung und den dazugehörigen Testmaßnahmen. Die folgende Abbildung zeigt den grundlegenden Aufbau des V-Modells.

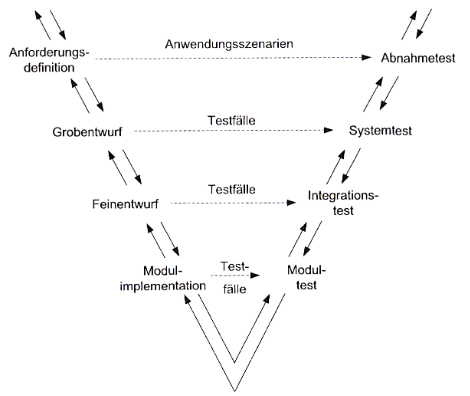


Abbildung 1: V-Modell nach (Diesterer et al., 2003) S.543

Auf der linken oberen Seite des V-Modells beginnt die Entwicklung mit den Anforderungen des zu entwickelnden Systems. Je tiefer man geht desto detaillierter werden die Anforderungen und desto näher kommt man zur Implementierung. An der Spitze des V-Modells ist das Ergebnis zu finden, also das fertige Produkt. Auf der rechten Seite wird diese Implementierung entsprechend den links vorgenommenen Spezifikationen getestet. Durch die Testfälle lassen sich phasenbezogene Ergebnisse für die darunter liegende Projektphase abheben.

Neue Softwareentwicklungsansätze machten eine Überarbeitung des V-Modells notwendig. Das V-Modell ist allgemeingültig verfasst und als Ergebnis wurde das V-Modell 97[[2]](#footnote-2) veröffentlicht. Im Zuge von neuen Erkenntnissen in der Softwareentwicklung wurde das V-Modell 97 durch das aktuelle V-Modell XT[[3]](#footnote-3) erfolgreich abgelöst.

Die in dem V-Modell enthaltenen vier Submodelle:

* Systemerstellung (SE)
* Qualitätssicherung (QS)
* Konfigurationsmanagement (KS)
* Projektmanagement (PM)

werden im Folgenden nach den Vorgaben des V-Modells XT auf die lösende Aufgabe bezogen betrachtet.

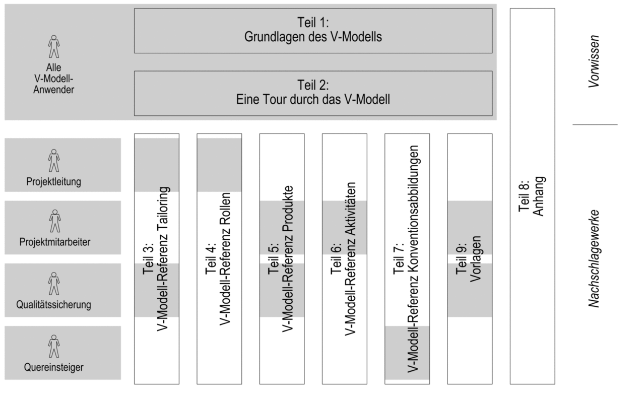


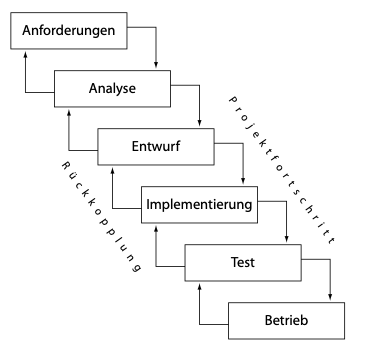
Abbildung 2: Aufbau des V-Modell XT 1.3 nach (IT-Beauftragte der Bundesregierung, 2006)

Das V-Modell hat folgende Zielsetzungen:

* Minimierung der Projektrisiken
* Verbesserung und Gewährleistung der Qualität
* Eindämmung der Gesamtkosten über den gesamten Projekt- und Systemlebenszyklus
* Verbesserung der Kommunikation zwischen allen Beteiligten

Das V-Modell definiert Rollen und Anforderungen an die Entwicklungsprozesse für komplexe Projekte. Für die Abwicklung öffentlicher Aufträge ist das V-Modell Standard in Deutschland und ist über das Bundesverwaltungsamt verfügbar. Die „Entwicklungsstandards für IT-Systeme des Bundes“ wurden auf Basis des V-Modells entwickelt. Das V-Modell hat sich zum Industriestandard entwickelt und zahlreiche Normen beeinflusst, die Entwicklungsprozesse adressieren. Somit ist das V-Modell XT kompatibel zu Standards wie ISO 9001 (Deutsche Norm, 2008b), CMMI, und SPICE.

## Wasserfall-Modell



Beim Wasserfallmodell wird sequenziell vorgegangen. Wenn eine Phase vollständig abgeschlossen wurde ist das Ergebnis dieser Phase die Voraussetzung für die nächstbeginnende Phase. Sollte eine Phase nicht den Anforderungen entsprechen so muss diese Phase erneut durchlaufen werden.

## Agiles Vorgehen – SCRUM

### SCRUM-Allgemein

Scrum (=aus englisch für „[das] Gedränge“) gehört zu den sogenannten Agilen Prozess Methode. Im Gegensatz zu den Klassischen/Definierten Prozess Methoden, welche zur Annahme keine unbekannten Faktoren und einen definierten Plan pflegen, sind die agilen Methoden flexibler.

Scrum ist ein „Vorgehensmodell mit Meetings, Artefakten, Rollen, Werten und Grundüberzeugungen, das beim Entwickeln von Produkten im Rahmen agiler Softwareentwicklung hilfreich ist“[[4]](#footnote-4).

### SCRUM-Rollen

Neben verschiedenen Rollen, wie beispielsweise Beobachter, Experten und „Stakeholdern“, sind die drei wichtigsten Rollen der Product-Owner, der Scrum-Master und das Scrum-Team.

Der Product-Owner vertritt den Auftraggeber, der fachliche Ansprüche vorgibt, die dann priorisiert werden. „Der Scrum-Master koordiniert das Team und ist für das Prozessmanagement zuständig, indem er dem Team den Rücken freihält und etwaige Hindernisse abklärt und aus dem Weg räumt“[[5]](#footnote-5). Das Scrum-Team ist eine heterogene Gruppe und nicht hierarchisch gegliedert und entwickelt das Produkt. Jedes Mitglied ist auf ein anderes Gebiet spezialisiert, aber dennoch in verschiedene Fachrichtungen ausgebildet und kann somit auch die Aufgaben von anderen Team-Mitgliedern übernehmen.

### SCRUM-Ablauf

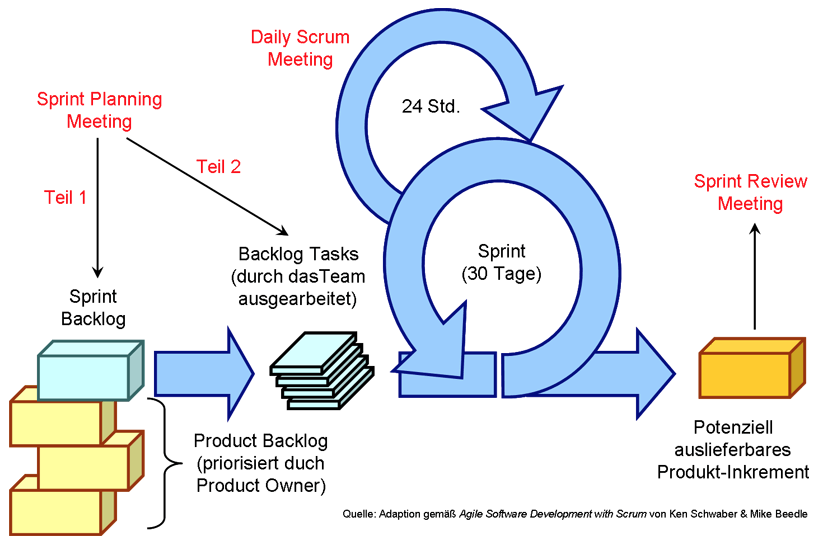
Der erste Schritt, um ein Projekt starten zu können, ist dass man die verschiedenen Anforderungen (Requirements) definiert. Diese werden nach Wichtigkeit priorisiert und in einer Liste gepflegt. Man nennt diese Liste das „Product-Backlog“. Das „Product-Backlog“ verändert sich laufend. Hierbei ist die Aufgabe des „Product-Owners“, die Arbeitspakete neu zu priorisieren und laufende Anforderungen in Hinblick auf die Wünsche des Kunden anzupassen.

„Damit ein produktives Arbeiten möglich ist, wird in Kooperation mit dem Product-Owner ein definiertes Arbeitspaket ausgewählt und in ein fertiges und auslieferbares Teilprodukt – das „Increment-of-Potentially-Shippable-Functionality“ – inklusive Dokumentation und Tests umgesetzt“[[6]](#footnote-6). Anders ausgedrückt bedeutet das, „Das Increment wird innerhalb der laufenden Iteration, also einem „Sprint“, gemäß der vorherigen Definition fertiggestellt. Eine zusätzliche Modifikation durch geänderte oder zusätzliche Anforderungen ist innerhalb der Sprint-Phase nicht erlaubt“[[7]](#footnote-7).

Wie man es auch beim klassischen Projektmanagement kennt, wird ein Arbeitspaket in kleinere Tasks eingeteilt. Diese Tasks werden im Sprint-Backlog dokumentiert und jeden Tag aktualisiert. Um das Umsetzen zu können müssen Meetings abgehalten werden. Damit man den Überblick nicht verliert, wann und wie welches Increment umgesetzt wird, werden täglich „Daily-Scrum-Meetings“ abgehalten, die etwa 15 Minuten andauern. Dabei können sich die Entwickler über mögliche Probleme und den aktuellen Entwicklungsstand beraten und austauschen.

Am Schluss eines Sprints, das zwischen 2-6 Wochen lang dauert, stellt das Scrum-Team dem Product Owner, den Stakeholders u.a. in einem **Sprint Review Meeting** direkt am System die neue entwickelte Funktionalität. Halbfertige Teilprodukte oder nur Powerpoint-Folien sind während des Sprint Reviews nicht erlaubt. Das Feedback der Stakeholder und die neuen Requirements des Product Owners für den nächsten Sprint fließen dann in das nächste Sprint Planning Meeting ein, und der Prozess beginnt von vorne[[8]](#footnote-8).

„Im Kern basiert Scrum also auf einer inkrementellen Vorgehensweise, der Oganisation von Entwicklungsabschnitten und Meetings in vordefinierten Zeitabschnitten (Time-Boxes) und der Erkenntnis, dass ein funktionierendes Produkt wichtiger ist als eine dreihundertseitige Spezifikation[[9]](#footnote-9).“



# Zu beurteilende Softwarekriterien

## Softwarekosten

Um eine Software zu erstellen entstehen viele Kosten wie:

* Rechercheaufwand
* Installation der Software
* Anpassungen
* Schulungen für das Personal
* Pflege und Wartung
* Updates
* Wartungskosten
* Mitarbeiterkosten
* Hardware, Strom und Internet

## Technische Anforderungen der Software

Die technischen Anforderungen spielen eher eine geringe Rolle in der Bewertung, da eine Hardware sehr preiswert ist und leicht ausgetauscht oder erweitert werden kann.

Neben Hardwarekosten entstehen aber auch noch Nebenkosten wie Dokumentationsänderungen, Inkompatibilität der bereits laufenden Hardware sowie Schulungskosten.

## Unternehmensspezifische Anforderungen der Software

Unternehmensspezifische Anforderungen sind nicht technische Anforderungen wie:

* Datenschutzrichtlinien
* Steuerregelungen
* Normen
* Risikomanagement

## Benutzeroberfläche

Zur Bewertung der graphischen Benutzeroberfläche sind die notwendigen Anforderungen:

* **Aufgabenangemessenheit:** Dem Benutzer sollen nur die wichtigsten Informationen, im Zusammenhang mit der Aufgabe, angezeigt werden
* **Selbstbeschreibungsfähigkeit:** auf den ersten Blick einschätzen welches grafische Element welche Funktion hat
* **Steuerbarkeit:** Sind Steuerelemente erkennbar die einem die Navigation innerhalb eines Arbeitsablaufs ermöglicht. Gibt es Vor-/Zurück Buttons. Der aktuelle Arbeitsschritt soll innerhalb der Prozessdarstellung hervorgehoben werden
* **Erwartungskonformität:** Bedienelemente sollen dort vorhanden sein wo sie zu erwarten sind. Funktionen die man häufiger benutzt leicht wiederzufinden

## Softwaredokumentation

Die Softwaredokumentation dient als Hilfe für die Benutzer. Es sollte prozessorientiert strukturiert und leichten und schnellen Zugang zur gesuchten Information darstellen.

Sie besteht aus:

* Programmierdokumentation (Quellcodebeschreibung)
* Methodendokumentation
* Installationsdokumentation
* Benutzerdokumentation (Handbuch, Online-Hilfe etc.)
* Datendokumentation
* Testdokumentation
* Entwicklungsdokumentation

## Datensicherung

Datensicherung heißt, dass man die Daten vor Verlust, Manipulation oder vor unberechtigten Zugang durch Dritte schützt. Bei der Datensicherung werden die Daten kopiert. (Backup)

## Datenschutz

Datenschutz ist die Vertraulichkeit, Verfügbarkeit und Integrität von Daten in Softwaresystemen. Bei der Bewertung von Software ist der Datenschutz und die damit verbundenen Informationssicherheit ein wichtiger Aspekt.

## Schnittstellen

Bei der Bewertung von Schnittstellen stellt sich die Frage welche Schnittstellen man für die Software hat.

Allgemein wird zwischen Daten-, Maschinen-, Hardware-, Netzwerk-, Software- und Benutzerschnittstellen unterschieden.

## Performance

Für die Performance ist die Geschwindigkeit, Reaktionszeit, Bearbeitungszeit und Berechnungszeit entscheidender Faktoren bei der Bewertung. Eine langsame Software kommt bei den Benutzern seht schlecht an

# Prozessstandards ISO 12207 und ISO 15288

## Softwareentwicklungsprozess

Bei der Entwicklung einer Software ist es wichtig systematisch vorzugehen, wobei der Prozess durch Vorgehensmodelle strukturiert wird. Die Vorgehensmodelle beinhalten die Organisation für die Softwareentwicklung sowie eine gut durchdachte Strategie für die Durchführung eines Projekts.

Dazu gehört:

* Das Erstellen einer Aktivitäten Liste, wo die Reihenfolge des Vorgehens enthalten ist
* Die Festlegung des Inhalts und der Form von Teilprodukten
* Die Kriterien für die Fertigstellung und Abnahme der Software
* Die Verteilung von Rollen, Verantwortlichkeiten sowie Kompetenzen
* Standards, Richtlinien, Methoden & Werkzeuge

Die Aufgabe dieser Vorgehensmodelle ist es den komplexen Prozess der Entwicklung und der anschließenden Wartung in überschaubare Teilaktivitäten zu zerlegen und deren Ergebnisse sowie logische und zeitlichen Zusammenhänge zu definieren.

Diese Gliederung wird anhand der folgenden Normen dargestellt.

## Systemlebenszyklus ISO 15288

Die ISO/IEC-Norm 15288 ist eine international aufgestellte Norm mit der Zielstellung der korrekten Bewertung und Verbesserung von Prozessen innerhalb eines Systems. Als System wird hierbei eine Kombination aus miteinander interagierenden System-Elementen wie z.B. Hardware, Software, Daten, menschliches Kapital, Prozesse oder Materialien bezeichnet. Jedes dieser einzelnen System-Elemente könnte aber wiederum als eigenes System betrachtet werden. Ein Beispiel hierfür wäre die ISO/IEC-Norm 12207, die im nächsten Punkt nochmals genauer unter die Lupe genommen wird. Ebenso basiert die Terminologie der Prozesse auch auf der ISO/IEC-Norm 12207.

Die einzelnen System-Elemente sind bei einfacheren Systemen meist hierarchisch, werden bei den neusten, komplexeren Systemen aber eher als Netzwerke aufgebaut. Außerdem gibt es keinen Standard für ein System, denn jedes System ist auf die aktuelle Situation und die definierten Grenzen dieses Systems zugeschnitten (Tailoring). Die ISO/IEC 15288 stellt lediglich einen Rahmen aller Prozesse dar, die je nach Bedarf in Betracht zu ziehen sind.

Die Themen Qualität und Projektmanagement werden hier nur angerissen, da es hierfür eigene Standards gibt.

## Softwarelebenszyklus ISO 12207

Die ISO/IEC-Norm 12207 ist ebenfalls eine international aufgestellte Norm, mit dem Ziel ein besseres Verständnis der Produktion von Software und deren zugehörigen Serviceleistungen. Wie schon in ISO/IEC 15288 erwähnt ist die Terminologie der Prozessstruktur fast gleich aufgebaut und wird im Folgenden näher erläutert:

Ein Bild, das Screenshot, Text enthält.

Automatisch generierte Beschreibung

Wie in Abb. 3.2.1 dargestellt, unterteilen sich die Lebenszyklus-Prozesse in folgende Gruppierungen:

**Agreement Processes:** Diese Prozesse beinhalten die Akquisition und die Lieferung. Dabei wird analysiert welche Services von externen Anbietern herangezogen werden müssen, um die Anforderungen des Projektes zu erfüllen. Der Supply-Prozess befasst sich mit dem Ausliefern von Services an Dritte. Dabei steht der Erwerber und Lieferant im Dialog. Das Ergebnis ist eine vertragliche Regelung, in dem Vereinbarungen getroffen werden.

**Organizational Project-Enabling Processes:** Hier wird sichergestellt, dass alle möglichen Ressourcen, die für das Umsetzen des Projektes benötigt werden, auch zur Verfügung gestellt werden. Zu diesen Ressourcen zählen:

- Entwicklungsumgebung,

- Infrastruktur (hinsichtlich Hardware, Server, etc.),

- humane Ressourcen (Personal: Entwickler, Manager, etc.)

Des Weiteren wird hier sichergestellt, dass gegebene Ressourcen, wie Wissen (Knowledge Management) und finanzielle Mittel des Projektes in Betracht gezogen werden, um möglichst kostenminimal zu arbeiten. Während des Software-Life-Cycles werden alle möglichen Qualitätsmaßnahmen innerhalb und außerhalb der Implementierung in Betracht gezogen, die zur Erfüllung der Zielvorgaben und damit zur Zufriedenstellung des Kunden führen

**Technical Management Processes:** Das technische Management sieht sich in seiner Aufgabe ähnlich wie die organisierenden Prozesse. Dabei wird der Hauptfokus auf das Management der technischen Umsetzung des Projektes gelegt. Es werden Projektpläne erstellt, die aufeinander abgestimmt und koordiniert werden. Zudem wird der aktuelle Status des Projektes überwacht, hinsichtlich Performanz, zeitliche Abstimmungen und Einhaltung von definierten Vorgaben. Um flexibel auf Probleme zu reagieren wird ein strukturiertes, analytisches Framework erschaffen, bei dem Entscheidungsalternativen in Betracht gezogen werden, die sich meist auf strategische Schritte beziehen. Außerdem werden Maßnahmen zur Risikobewältigung während des Entwicklungsprozesse getroffen, dazu wird sichergestellt, dass Maßnahmen aus dem Qualitätsmanagement auf das Projekt angewendet werden.

**Technical Processes:** Zu guter Letzt werden hier die rein technischen Prozesse beleuchtet. Das Ziel besteht darin, aus der Anforderungsanalyse eine best mögliche technische Umsetzung in die Wege zu leiten. Dazu wird zuallererst analysiert welche Anforderungen der Kunde an das Projekt hat. Daraus wird dann herausgearbeitet, welche Anforderungen nun das System bzw. einzelne Systemelemente erfüllen muss, um die vordefinierten Zielvorgaben zu erfüllen.

Folgende Prozesse werden nun in die Wege geleitet:

1. Die System Architektur, auf der das Projekt basieren soll, wird so ausgewählt bzw. verändert, dass die Anforderungen best möglich erfüllt werden können.
2. Es werden alle essentiellen Informationen und Daten bereitgestellt, damit die Implementierung mit den Architekturelementen konsistent gehalten werden können. Während der kompletten Implementierungsphase werden wichtige Daten wie technische Performanz, System-/Softwareverhalten und Risiken überwacht und dokumentiert.
3. Auf Basis der vorigen Prozesse erfolgt die Implementierung
4. Nach dem Fertigstellen von einzelnen Systemelementen, werden diese in den Gesamtkontext des Projektes gebracht und eingebunden. Der Verification Prozess prüft dann einzelne Elemente auf ihre Erfüllung der Vorgaben
5. Der Transition-Prozess wird eingeleitet, sobald die Software lauffähig ist und macht dieses operabel, kompatibel und funktional, sodass das Produkt als Ganzes ausgeliefert werden kann
6. Im darauffolgenden Validierungsprozess prüft der Stakeholder, ob das fertige Produkt die vereinbarten Forderungen erfüllt
7. Nach dem Etablieren des Projektes wird die Dauerhaftigkeit und Funktionalität des Programms überwacht und auf Fehler Reports seitens des Kunden eingegangen
8. Zuletzt folgt der Disposal Prozess, der die Existenz von einzelnen Softwareelementen (oder im Extremfall die komplette Software) beendet und eventuelle Änderungen vornimmt.

\_\_\_

Abbildung : Softwarelebenszyklus

1. **Anforderungsanalyse:** In der Anforderungsanalyse werden die Ziele der Softwareentwicklung beschrieben. Sie formuliert die funktionalen und nicht-funktionalen Anforderungen.
2. **Systemspezifikation:** Hier werden mit Hilfe von Analysetechniken der Ist – Zustand, Schwachstellen, fachliche Beschreibung von Funktionalität und Leistungsumfang festgehalten. Als Ergebnis wird ein Pflichtenheft formuliert.
3. **Modellierung:** Das zu konstruierende System wird als Entwurf modelliert. Der Entwurf enthält die Struktur des Systems, wie es die Funktionen erfüllt und wie es sich zur Laufzeit dynamisch verhält.
4. **Implementierung:** Bei der Implementierung wir der Systementwurf in eine Software umgesetzt. Das Ergebnis sind Programme und Dokumentation.
5. **Integration & Test:** Hier werden alle Komponenten zu einem Gesamtsystem zusammengeführt und getestet. Daraus entsteht das auslieferfähige Endprodukt.
6. **Betrieb & Wartung:** Bei Betrieb und Wartung werden Fehler korrigiert und es entstehen neue Anpassungen weshalb der Kreislauf von vorne beginnt.

\_\_\_

Die ISO IEC 12207 adressiert Anforderungen an den Softwarelebenszyklus und definiert folgende Prozesse:

1. Einigungsprozesse
2. Organisatorische Prozesse zur Projektunterstützung
3. Projektbezogene Prozesse
4. Technische Prozesse
5. Software-Entwicklungsprozesse
6. Software-Unterstützungsprozesse
7. Software-Wiederverwendungsprozesse

Ähnlich wie beim V-Modell werden folgende Inhalte adressiert: Dokumentation, Konfigurationsmanagement, Qualitätssicherung, formale Überprüfung der Prozesse (Verifikation), inhaltliche Überprüfung (Validierung), Review, Audit und Problembehandlung.

# Softwareprozessverbesserungs- und Bewertungsmodelle – CMMI/SCAMPI, ISO 15504/ISO IEC 330xx-Normenreihe

Reifegradmodelle dienen der qualitativen Bewertung von Softwareentwicklungsprozessen. Je höher der Reifegrad einer Entwicklungsorganisation ist, desto höher ist die Wahrscheinlichkeit einer qualitativ hochwertigen und fehlerarmen Software. Im Hinblick auf die Praxis werden Reifegradbewertungen bei Lieferanten und Entwicklungsorganisation angewendet. Die gängigen Reifegradmodelle werden im Folgenden eingeführt.

## CMMI (Capability Maturity Model Integration)

CMMI[[10]](#footnote-10) ist ein Modell zur Reifegradbewertung von Entwicklungsprozessen und unterscheidet sich somit von den Prozessmodellen wie der ISO IEC 12207. Es dient maßgeblich zur Bewertung der Prozesslandschaft, aber auch zur Identifikation von Verbesserungspotentialen. Zur Bewertung werden folgenden Kriterien angewendet:

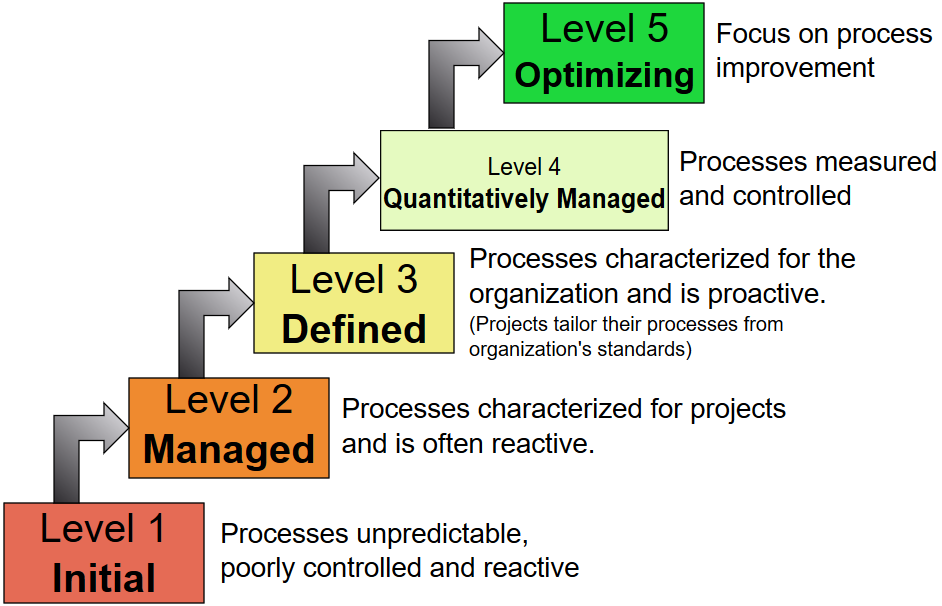


Abbildung 4: Charakteristik der Reifegrade

Für die einzelnen Bewertungslevel gelten folgenden Kriterien:

* **Level 1 – Initial:** 
  + Keine Kontrollen, keine Anforderungen. Diesen Reifegrad hat jede Organisation automatisch.
* **Level 2 – Managed:**
  + Die Projekte werden geführt. Ein ähnliches Projekt kann erfolgreich wiederholt werden.
* **Level 3 – Defined:**
  + Die Projekte werden nach einem angepassten Standardprozess durchgeführt und es gibt eine organisationsweite kontinuierliche Prozessverbesserung.
* **Level 4 – Quantitatively Managed:**
  + Es wird eine statistische Prozesskontrolle durchgeführt.
* **Level 5 – Optimizing:** 
  + Die Arbeit und Arbeitsweise werden mit Hilfe einer statistischen Prozesskontrolle verbessert.

Die Reifegrade sind in allen CMMI Modellen grundsätzlich identisch aber die Zuordnung der Prozessgebiete zu den fünf Reifegraden ist spezifisch für jedes CMMI-Modell.

Die Bewertung des Reifegrades bzw. des Fähigkeitsgrads einer Organisation geschieht durch eine **SCAMPI-**Untersuchung, die nur durch vom SEI[[11]](#footnote-11) autorisierte Personen geleitet werden kann.

Man unterscheidet drei verschiedene Varianten von CMMI-Modellen. CMMI für Entwicklung (CMMI-DEV) wird für die Entwicklung von Produkten oder für Wartungsprojekte zu existierenden Produkten verwendet. Das CMMI for Services (CMMI-SVC) wird für Organisationen verwendet, die Dienstleistungen anbieten. CMMI-SVC adressiert alle Arten von Dienstleistungsorganisationen. Für IT-Betriebsorganisationen stellt CMMI for Services eine Alternative zu ITIL dar. Im Vergleich zu ITIL ist CMMI for Services höher aggregiert. CMMI for Services und CMMI for Development können miteinander integriert werden, so dass sie zusammen den gesamten Produkt-Lifecycle abdecken.

Um CMMI erfolgreich verwenden zu können, ist es zwingend notwendig ein konkretes Verbesserungsziel zu verfolgen. CMMI hilft, gezielt die relevanten Praktiken durchzugehen, bei denen die Frage im Vordergrund steht, ob diese Praktiken im Sinne des Verbesserungsziels unter Kontrolle sind oder ob eine Optimierung sinnvoll erscheint. Ohne ein Verbesserungsziel kann eine Organisation nur zufällig Verbesserungen erreichen.

**Abgrenzung zu anderen Normen:**

Im Unterschied zu Qualitätsmanagementnormen wie DIN EN ISO 9001 gehen die CMMI-Modelle spezifisch auf die Praktiken in einem bestimmten Anwendungsgebiet ein. Während DIN EN ISO 9001 die gesamte Organisation und damit mehr die Breite abdeckt, geht CMMI bei den konkreten Tätigkeiten weit mehr in die Tiefe und bietet konkrete Prozessgebiete und Praktiken.

Die CMMI-Modelle setzen die Anforderungen der Norm ISO/IEC 15504 (SPICE) an ein Prozessmodell um. Neben den CMMI-Modellen gibt es auch die Prozessmodellnormen ISO/IEC 12207 für Software- und ISO 15288 für die Systementwicklung. Im Gegensatz zu CMMI gehen diese beiden Normen aber nicht über die Definition der Titel der Praktiken von CMMI hinaus, d.h. es wird nicht umfangreicher erklärt als es in CMMI bereits getan wird.

## ISO 15504 (SPICE) & ISO IEC 3300x-Normenreihe

Im Rahmen des Qualitätsmanagements kann die Bewertung, Betrachtung und Verbesserung von Prozessen über die sog. Reifegradbewertung und Prozessverbesserung erfolgen. Standards zu dieser Betrachtungsform wurden innerhalb der ISO-Richtlinie 15504 auf internationale Ebene spezifiziert und vorgeschrieben. Aufgrund der prozessbezogenen Bewertungsmöglichkeit (Assessment) und Aussagekraft wird diese ISO-Richtlinie auch als SPICE bezeichnet. Das Akronym steht für „**S**oftware **P**rocess **I**mprovement and **C**apability **D**etermination“. Der ISO Standard wird dabei in folgende zehn Bände aufgeteilt:

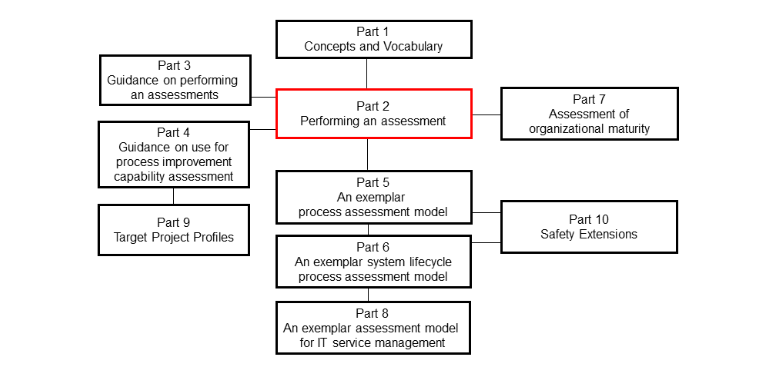


Abbildung 4.2 Interdependenzen zwischen ISO 15504 Bänder

Die Bände stellen den Aufbau der Normenreihe aus der ISO-Reihe IEC 15504 dar und sind untereinander interdependent zu sehen. Das wesentliche Ziel der Norm stellt die Verbesserung von Prozessen in den Mittelpunkt. Die Methodik zur Prozessbeurteilung (engl. Capability Determination) und Prozessverbesserung (Process Improvement) erfolgt anschließend beispielsweise über das in dieser Ausarbeitung behandelte Reifegradmodell.

Die Richtlinien beziehen sich in ihrer Standardisierung auf die Themenbereiche der Arbeitsorganisation, Methodik und Benutzerorientierung der Prozesse (hier: Softwareentwicklung als Entwicklungsprozess). Beim Übergang zu den im Jahre 2015 überarbeiteten Richtlinien, die als Fortführung der ISO IEC 155xx Normenreihe nun als ISO IEC 330xx beziffert werden, zeigen sich als wesentliche Änderungsmerkmale die Ausarbeitung des SPICE-Konzepts für den Automotive-Bereich, sowie eine ausgeweitete Reifegradbewertung der Prozesse.[[12]](#footnote-12)

Das SPICE Konzept teilt sich für die Prozessbeurteilung sowie -verbesserung in folgende wesentliche Bestandteile und Kriterien auf:[[13]](#footnote-13)

* Process Assessment Model (PAM)
  + Capability Levels (Fähigkeitsdimension)
  + Process Attributes
  + Generic Practices
  + Generic Work Product
* Process Reference Model
  + Process Areas
  + Processes Base Practices
  + Work Products

Das Process Reference Model geht hierbei aus der ISO Richtlinie IEC 12207 hervor und ist nach Primär-, Sekundär- und Organisationsprozessen gegliedert.[[14]](#footnote-14) Innerhalb der SPICE-Assessments sollen Prozessänderungen als Abhängigkeit zu den Faktoren Person und Produktqualität gesehen werden. Prozessänderungen, und somit auch Prozessverbesserungen, sollen innerhalb der Prozesslandschaft nicht isoliert betrachtet werden.

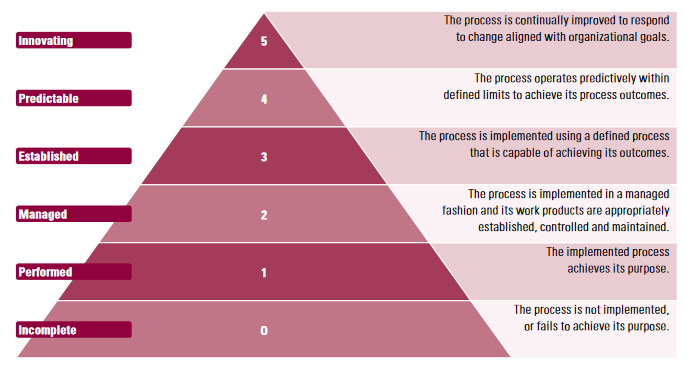


Abbildung 4.3 SPICE Capability Levels[[15]](#footnote-15)

Im Assessment Model (PAM) dienen die in der obigen Abbildung aufgeführten Fähigkeitsdimensionen zur Einstufung und Einschätzung der Prozesskapazität. Daraus lassen sich Potenziale ermitteln, die im Rahmen der Prozessverbesserung vorgenommen werden können. Weitere Bewertungsmöglichkeiten sind im Folgenden aufgeführt.

Andreas Baerwald, Uni Regensburg.  
<https://epub.uni-regensburg.de/38145/1/Dissertation_Andreas_Baerwald_2018-12-17_final.pdf>

SE-Konferenzen  
<http://www.se-konferenzen.de/bisher/se2007/folien/se2007-B2-3.pdf>

Kugler Maag  
<https://www.kuglermaag.de/fileadmin/05_CONTENT_PDF/2-10_automotive-spice_version_3_pocketguide.pdf>

## Die Bewertungsmethode SCAMPI

# Anwendung am Fallbeispiel – Projektstudium SAPlexa

## Projektbeschreibung- & Ziel

Als Fallbeispiel sollen in dieser Ausarbeitung die Entwicklungsprozesse aus dem vorangegangenen Projektstudium hinzugezogen und kritisch im Hinblick auf die behandelten ISO-Standards zur Bewertung von Entwicklungsprozessen beurteilt werden. Das Projekt SAPlexa beschäftigt sich mit einer Einbindung einer Sprachassistenz in einem Geschäftsbereich bzw. einem Teilprozess des Wareneingangs. Die Entwicklung von SAPlexa schließt eine eigens hierfür zu entwickelnde GUI-Oberfläche ein, welche unterstützend zur Sprachsteuerung zu verstehen ist. Durch adäquate Schnittstellenprogrammierung sollen in diesem Zuge Datenströme zwischen der Java-basierten GUI und der SAP HANA Datenbank auf das Nötigste reduziert und für den Nutzer (hier: Lagerist) visualisiert werden.

Im Vordergrund steht hierbei die effizientere Bedienung des SAP Programms über dedizierte Sprachbefehle, die den Einlagerungs- und Wareneingangsprozess ergonomisch und zeitsparend gestalten sollen. Die Implementierung und Entwicklung sollen zudem die speziellen Anforderungen aus dem Fachbereich der Logistik, somit die geschäftsprozessgebundenen Anforderungen an die Software, berücksichtigen. Inkrementell soll daraus ein lauffähiges Programm entstehen, welches die persistente Buchung eines Wareneingangs technisch zuverlässig im SAP System vornehmen kann.

## Vorgehensweise

Für die Vorgehensweise im Fortgang des Projekts standen die Prinzipien eines SCRUM-Teams aus der agilen Softwareentwicklung im Mittelpunkt. Regelmäßige SCRUM-Meetings dienten der zuverlässigen Erfassung des aktuellen Status und Fortschritts. Durch Festlegung unterschiedlich gewichteter (Implementierungs-) Aufgaben im Product-Backlog wurden jegliche Anforderungen vom Kunden an das Projekt festgehalten. Eine realistische, zeitliche Aufwandsabschätzung der einzelnen Aufgaben wurde zur Grundlage für eine solide und bindende Zeitplanung der SCRUM-Fristen. Die konkrete Festlegung einer Definition-of-Done im Sinne der SCRUM Prinzipien bewirkte eine zielführende Softwareentwicklung.

Für die Realisierung des Projekts standen die Anforderungen aus dem Geschäftsleben im Vordergrund. Die durchweg bestehende Benutzerorientierung – in Kooperation mit der Groz-Beckert KG – ermöglichte uns das Erfassen von Wünschen und praktischen Anforderungen, denen anfänglich im Projekt keine hinreichende Relevanz zugesprochen wurde.

* benutzerbezogen
* Anforderungen vom Kunden eingegangen (Product Owner) 🡪 Product BackLog
* agile Vorgehensweise, Sprint Reviews, Definition of Done schließt eine Phase ab bzw, startet nächsten Sprint, Bewertungspunkte (30 pro Woche)

## Kritische Betrachtung

* nicht gezielt nach Vorgehensmodell gearbeitet aber viele Punkte überschneiden sich mit V-Modell XT

# Fazit

Was wurde herausgefunden?

Zähle die wichtigsten [Ergebnisse](https://www.scribbr.de/aufbau-und-gliederung/forschungsergebnisse-bachelorarbeit/) deiner Forschung auf und erkläre, zu welchem [Fazit](https://www.scribbr.de/aufbau-und-gliederung/fazit-bachelorarbeit/) du gekommen bist

Was bedeuten deine Ergebnisse?

Nenne die relevantesten Eckpunkte aus der fachlichen [Diskussion](https://www.scribbr.de/aufbau-und-gliederung/diskussion-bachelorarbeit/) und lege deine [Empfehlungen](https://www.scribbr.de/aufbau-und-gliederung/handlungsempfehlung/) dar.

# Quellen- und Literaturverzeichnis

1. Bröhl & Dröschel, 1995 [↑](#footnote-ref-1)
2. (Juni 1997) [↑](#footnote-ref-2)
3. (Februar 2012) [↑](#footnote-ref-3)
4. Zitat([de.slideshare.net/6footplus/scrum-prsentation?qid=04836422-d9d7-455b-b415-d768907799bc&v=&b=&from\_search=1](https://de.slideshare.net/6footplus/scrum-prsentation?qid=04836422-d9d7-455b-b415-d768907799bc&v=&b=&from_search=1), 2020) [↑](#footnote-ref-4)
5. Zitat ([t3n.de/news/scrum-was-ist-das-506705/](https://t3n.de/news/scrum-was-ist-das-506705/), 2020) [↑](#footnote-ref-5)
6. Zitat ([t3n.de/news/scrum-was-ist-das-506705/](https://t3n.de/news/scrum-was-ist-das-506705/), 2020) [↑](#footnote-ref-6)
7. Zitat ([t3n.de/news/scrum-was-ist-das-506705/](https://t3n.de/news/scrum-was-ist-das-506705/), 2020) [↑](#footnote-ref-7)
8. Vgl. ([scrum-master.de/](https://scrum-master.de/),2020) [↑](#footnote-ref-8)
9. Zitat ([scrum-master.de/](https://scrum-master.de/),2020) [↑](#footnote-ref-9)
10. (CMMI Product Team, 2010) [↑](#footnote-ref-10)
11. Software Engineering Institutes [↑](#footnote-ref-11)
12. Andreas Baerwald, Uni Regensburg. [↑](#footnote-ref-12)
13. SE-Konferenzen. [↑](#footnote-ref-13)
14. SE-Konferenzen. [↑](#footnote-ref-14)
15. Kugler Maag. [↑](#footnote-ref-15)