

**Standards für die Bewertung**

**von Software-Entwicklungsprozessen**

**eingereicht von:**

Angelo Cavallaro

Beyza Gürler

David Süßlin

Hüseyin Kasarca

Nahom Ghebremussie

Viktor Sellmann

**eingereicht am:**

14. Februar 2020

**Begutachter:**

Herr Prof. Dr. German Nemirovskij

Herr Knut Kliem

Inhaltsverzeichnis

[Abbildungsverzeichnis 3](#_Toc32575252)

[Abstract 4](#_Toc32575253)

[1 Vorgehensmodelle in der Praxis 5](#_Toc32575254)

[1.1 RUP (Rational Unified Process) 5](#_Toc32575255)

[1.2 V-Modell XT 6](#_Toc32575256)

[1.3 Wasserfall-Modell 8](#_Toc32575257)

[1.4 Agiles Vorgehen – SCRUM 8](#_Toc32575258)

[1.4.1 SCRUM-Allgemein 8](#_Toc32575259)

[1.4.2 SCRUM-Rollen 8](#_Toc32575260)

[1.4.3 SCRUM-Ablauf 9](#_Toc32575261)

[2 Software-Beurteilungskriterien 11](#_Toc32575262)

[3 Prozessstandards ISO 12207 und ISO 15288 13](#_Toc32575263)

[3.1 Softwareentwicklungsprozess 13](#_Toc32575264)

[3.2 Systemlebenszyklus ISO 15288 14](#_Toc32575265)

[3.3 Softwarelebenszyklus ISO 12207 15](#_Toc32575266)

[4 Softwareprozessverbesserungs- und Bewertungsmodelle – CMMI/SCAMPI, ISO 15504/ISO IEC 330xx-Normenreihe 18](#_Toc32575267)

[4.1 CMMI (Capability Maturity Model Integration) 18](#_Toc32575268)

[4.2 Das Appraisal-Verfahren SCAMPI 20](#_Toc32575269)

[4.3 ISO 15504 (SPICE) & ISO IEC 3300x-Normenreihe 22](#_Toc32575270)

[5 Anwendung am Fallbeispiel – Projektstudium SAPlexa 24](#_Toc32575271)

[5.1 Projektbeschreibung- & Ziel 24](#_Toc32575272)

[5.2 Vorgehensweise 24](#_Toc32575273)

[5.3 Kritische Betrachtung 25](#_Toc32575274)

[Quellen- und Literaturverzeichnis 27](#_Toc32575275)

# Abbildungsverzeichnis

[Abbildung 1‑1: RUP 5](#_Toc32575276)

[Abbildung 1‑2: V-Modell nach (Riesterer et al., 2003) 6](#_Toc32575277)

[Abbildung 1‑3: Aufbau des V-Modell XT 7](#_Toc32575278)

[Abbildung 1‑4: Wasserfallmodell 8](#_Toc32575279)

[Abbildung 1‑5: SCRUM-Ablauf 10](#_Toc32575280)

[Abbildung 3‑1: Software-Lebenszyklus 13](#_Toc32575281)

[Abbildung 4‑1: Charakteristik der Reifegrade 19](#_Toc32575282)

[Abbildung 4‑2: SCAMPI Charakteristiken 21](#_Toc32575283)

[Abbildung 4‑3: Interdependenzen zwischen ISO 15504 Bänder 22](#_Toc32575284)

[Abbildung 4‑4: SPICE Capability Levels 23](#_Toc32575285)

# Abstract

Die Welt ohne Software ist in der heutigen Zeit unabdingbar. Für jedes noch so kleine Problem gibt es eine Software, die das Problem löst. Doch wie so eine Software entsteht und welcher 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 des Softwareentwicklungsprozesses zu untersuchen, analysieren und zu bewerten. Dazu betrachten wir unser Fallbeispiel aus unserem Projektstudium „SAPlexa“ mit der Problemstellung den Entwicklungsprozess kritisch zu analysieren und zu bewerten.

**Vorgehensweise**

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

# Vorgehensmodelle in der Praxis

Vorgehensmodelle dienen dazu, die Softwareentwicklung übersichtlicher zu gestalten und in der Komplexität beherrschbar zu machen. Da komplexe Software nur schwer zu erstellen und zu warten ist, bedienen sich Softwareentwickler eines Planes zur Entwicklung von Software. Diese Pläne sind die sog. Vorgehensmodelle. In den folgenden Absätzen werden die aktuellsten Vorgehensansätze in der Praxis vorgestellt.

## RUP (Rational Unified Process)

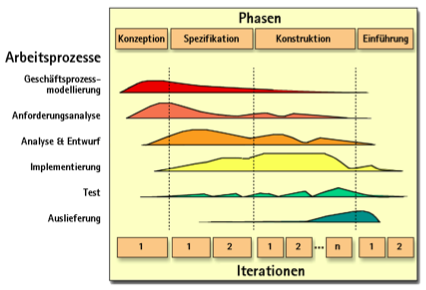
****

Abbildung ‑: RUP

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 beschreiben.

**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 folgen nur noch die Übergabephase und Auslieferung des fertigen Produktes an den Kunden.

## 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, das den typischen Entwicklungsprozess in definierte Phasen gliedert. Zusätzlich definiert das V-Modell das Vorgehen zur Qualitätssicherung in Form von Tests. Dadurch ist es möglich, sowohl methodisch als auch phasenorientiert an eine Softwareentwicklung vorzugehen und diese durch integrierte Testmaßnahmen qualitativ zu halten. Die folgende Abbildung zeigt den grundlegenden Aufbau des V-Modells.

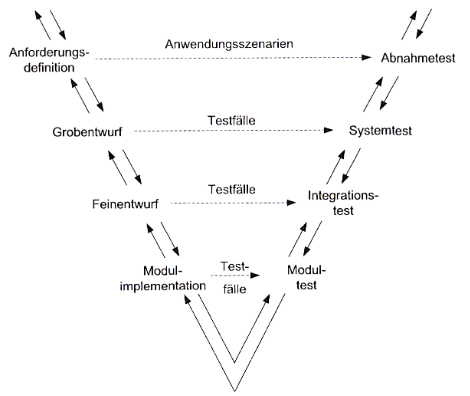


Abbildung ‑: V-Modell nach (Riesterer et al., 2003)

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 an das Implementierungsziel. An der Spitze des V-Modells ist das fertige Produkt als Ergebnis zu finden. Auf der rechten Seite wird diese Implementierung entsprechend den links vorgenommenen Spezifikationen getestet. Durch die Testfälle lassen sich phasenbezogene Ergebnisse für die darunter liegende Projektphasen abheben.

Durch neuere Softwareentwicklungsansätze, wie zum Beispiel das objektorientierte Programmieren, war es notwendig das klassische V-Modell zu überarbeiten. Als Ergebnis wurde das V-Modell 97 veröffentlicht. Im Zuge von neuen Erkenntnissen in der Softwareentwicklung wurde das V-Modell 97 durch das aktuelle V-Modell XT erfolgreich abgelöst. Folgende Punkte haben sich hierbei geändert:

* die Menge der Aktivitäten und Produkte können auf das notwendige Maß reduziert werden, um somit einen geringen Aufwand zu produzieren (Tailoring)
* der Auftraggeber ist nun Teil des Entwicklungsprozesses
* die vier Submodelle Systemerstellung, Qualitätssicherung, Konfigurationsmanagement und Projektmanagement werden durch Vorgehensbausteine abgelöst, die das Vorgehen definieren (s. nachfolgende Abbildung)
* der Fokus liegt mehr auf agilen und inkrementellen Ansätzen

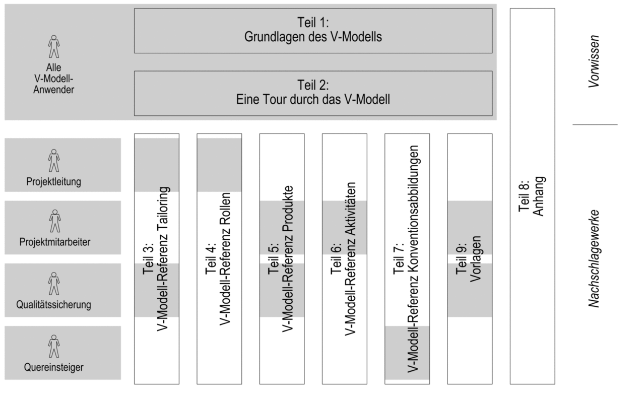


Abbildung ‑: Aufbau des V-Modell XT

Anhand der obigen Abbildung ist klar zu erkennen welche Rollen bzw. Teilbereiche im Projekt bestimmte Aufgabenteile zu verantworten haben.

Das V-Modell hat folgende Zielsetzungen:

* Projektrisiken sollen minimiert werden
* Qualität soll verbessert und gewährleistet werden
* Gesamtkosten über den gesamten Projekt- und Systemlebenszyklus sollen eingedämmt werden
* Kommunikation zwischen allen Beteiligten soll verbessert werden

Mittlerweile hat sich das V-Modell in der Industrie zu einem Standard entwickelt und zahlreiche Normen beeinflusst, die die Entwicklungsprozesse adressieren. Dadurch lässt sich das V-Modell XT mit anderen Standards, wie CMMI oder SPICE vereinbaren.

## Wasserfall-Modell

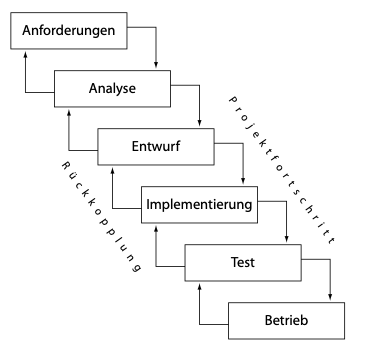


Abbildung ‑: Wasserfallmodell

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 Prozessmethoden. Im Gegensatz zu den klassischen/definierten Prozessmethoden, 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“[[1]](#footnote-1).

### 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“[[2]](#footnote-2). 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 das Definieren von verschiedenen Anforderungen (Requirements). 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 im 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“[[3]](#footnote-3). 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“[[4]](#footnote-4).

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.

Im Schluss eines Sprints, der 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[[5]](#footnote-5).

„Im Kern basiert SCRUM also auf einer inkrementellen Vorgehensweise, der Organisation von Entwicklungsabschnitten und Meetings in vordefinierten Zeitabschnitten (Time-Boxes) und der Erkenntnis, dass ein funktionierendes Produkt wichtiger ist als eine dreihundertseitige Spezifikation[[6]](#footnote-6).“

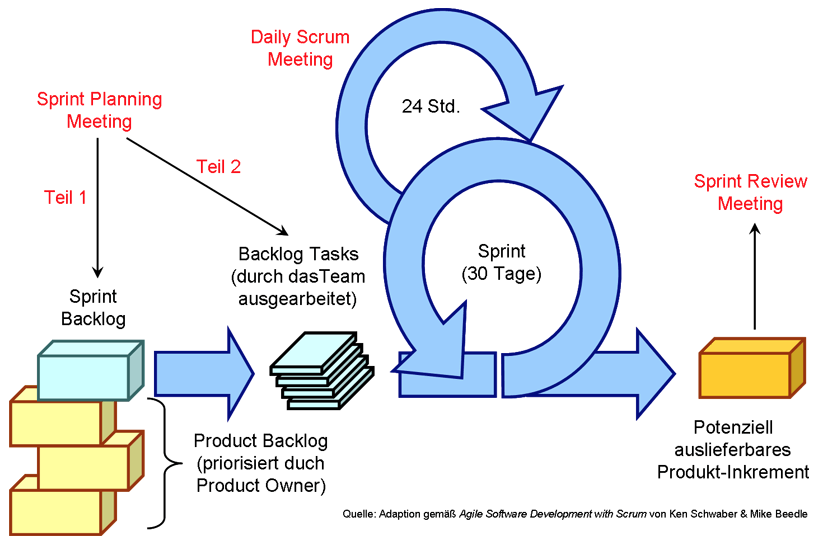


Abbildung ‑: SCRUM-Ablauf

# Software-Beurteilungskriterien

Um eine Software zu beurteilen, gibt es verschiedene Kritikpunkte, die zu beachten sind. Im unteren Teil wurden die wichtigsten Punkte aufgezählt

**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:** Die Benutzeroberfläche beschreibt seine Funktionen intuitiv. **Steuerbarkeit:** Es sind Steuerelemente erkennbar, die den Nutzer die Navigation innerhalb eines Arbeitsablaufs ermöglichen soll.
* **Erwartungskonformität:** Bedienelemente sollen dort vorhanden sein, wo sie zu erwarten sind. Ebenso sollen Funktionen, welche häufiger in Benutzung sind, leichter wiedergefunden werden.

**Softwaredokumentation**

Die Softwaredokumentation dient als Hilfe für den Benutzer. Sie sollte prozessorientiert und strukturiert aufgebaut sein sowie einen leichten und schnellen Zugang zur gesuchten Information darbieten.

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 sind die Geschwindigkeit, Reaktionszeit, Bearbeitungszeit und Berechnungszeit entscheidende Faktoren bei der Bewertung. Eine langsame Software kommt bei den Benutzern sehr 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, in der 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.

In der folgenden Darstellung wird der Softwareentwicklungsprozess bildhaft dargestellt und die einzelnen Phasen näher beleuchtet.

1 Anforderungs-analyse

2 Systemspezifikation

3 Modellierung

4 Implementierung

5 Integration & Test

6 Betrieb & Wartung

Abbildung ‑: Software-Lebenszyklus

Abbildung 3: 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 wird der Systementwurf in eine Software umgesetzt. Das Ergebnis sind Programme und Dokumentationen.
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.

Um einen Rahmen für die Regelungen und Vorgehensweisen zu schaffen, wurden hierfür Normen und Richtlinien in sogenannten ISO-Normen entwickelt. Die wichtigsten dieser Normen für den Softwareentwicklungsprozess werden nun noch etwas genauer dargelegt.

## Systemlebenszyklus ISO 15288

Die ISO/IEC-Norm 15288 ist eine international aufgestellte Norm mit der Zielstellung der 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

Abbildung 3.3 Life-Cycle-Prozesse

Wie in Abb. 3.3 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 stehen der Erwerber und Lieferant im Dialog. Das Ergebnis ist eine vertragliche Regelung, in dem Vereinbarungen getroffen werden.

**Organizational Project-Nübling 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 Entwicklungsprozesses 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 bestmögliche 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 Systemarchitektur, auf der das Projekt basieren soll, wird so ausgewählt bzw. verändert, dass die Anforderungen bestmöglich erfüllt werden können.
2. Es werden alle essenziellen 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 diese 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.

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

Um agile Software-Prozesse qualitativ bewerten zu können gehören sogenannte Reifegradmodelle zu den Standards für die Einschätzung der einzelnen Softwareentwicklungsprozessen. Je höher der Reifegrad einer Entwicklungsorganisation ist, desto höher ist die Wahrscheinlichkeit einer qualitativ hochwertigen und fehlerarmen Software. Die gängigen Reifegradmodelle werden im Folgenden eingeführt.

## CMMI (Capability Maturity Model Integration)

CMMI ist ein Modell zur Reifegradbewertung von Entwicklungsprozessen für unterschiedliche Anwendungsgebiete. Man unterscheidet drei verschiedene Varianten von CMMI-Modellen.

* CMMI for Development (**CMMI-DEV**) wird für die Entwicklung von Produkten oder für Wartungsprojekte zu existierenden Produkten verwendet.
* CMMI for Services (**CMMI-SVC**) wird bei der Etablierung und Lieferung von Dienstleistungen verwendet.
* CMMI for Acquisition (**CMMI ACQ**) bei der Beschaffung.

CMMI unterscheidet sich von Prozessmodellen wie der ISO IEC 12207, da der Fokus hierbei auf einer guten Produktentwicklung (das „Was“) statt des Definierens von konkreten Schritten (das „Wie“) liegt. Eine besondere Eigenschaft der CMMI-Modelle ist, dass sie nicht nur auf die fachlichen Praktiken eingehen, sondern auch auf die unterstützenden Aufgaben der Organisation wie zum Beispiel der Bereitstellung von Ressourcen oder die Durchführung von Trainingsmaßnahmen.

Ein weiterer nennenswerter Punkt der CMMI-Modelle ist die Institutionalisierung, d.h. dass die Arbeitsweisen in der Organisation als selbstverständlich und als Teil der täglichen Arbeit gelebt werden. Besonders in Zeiten von Stress und hohem Druck haben institutionalisierte Arbeitsweisen eine hohe Beständigkeit. Der Grad der Institutionalisierung wird durch sog. Fähigkeitsgrade wie folgt definiert:

* **Fähigkeitslevel 0 – Incomplete**
  + Die Arbeit wird so durchgeführt, dass die fachlichen Ziele nicht erreicht werden.
* **Fähigkeitslevel 1 – Performed**
  + Die Arbeit wird so durchgeführt, dass die fachlichen Ziele erreicht werden.
* **Fähigkeitslevel 2 – Managed**
  + Die Arbeit wird beobachtet und gelenkt.
* **Fähigkeitslevel 3 – Defined**
  + Die Arbeit wird in Form von einem Standardprozess definiert und angepasst. Dadurch wird die Arbeitsweise verbessert.

Neben den Fähigkeitslevel eines einzelnen Prozessgebietes, definiert CMMI Reifegrade, die durch folgende Kriterien bewertet werden:

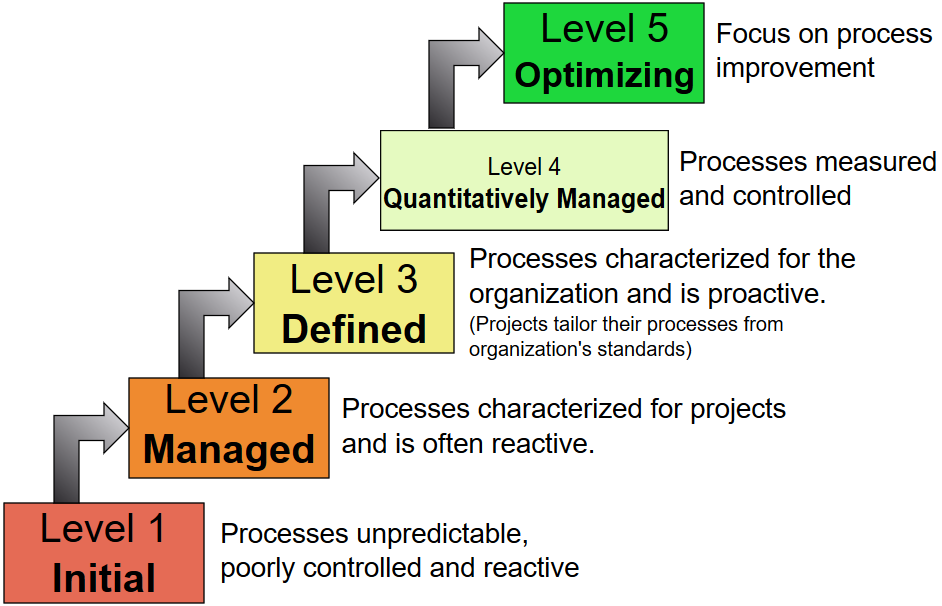


Abbildung ‑: Charakteristik der Reifegrade

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.

## Das Appraisal-Verfahren SCAMPI

SCAMPI (**S**tandard **C**MMI **A**ppraisal **M**ethodfor **P**rocess **I**mprovement) ist das Appraisal-Verfahren des Software Engineering Institutes (SEI) und dient dazu den Reifegrad bzw. den Fähigkeitsgrad einer Organisation zu bewerten. Appraisal (= Bewertung, Beurteilung) steht hierbei für die Überprüfung einer Organisation hinsichtlich der Umsetzung der Anforderungen eines Prozessmodells, wie beispielsweise den CMMI-Modellen oder der ISO/IEC 12207. Zum Beispiel ist es möglich gemeinsam mit dem Team zu ermitteln welche Prozessschritte eingeleitet/verbessert werden, müssen um einen höheren Reifegrad zu erreichen.

Man unterscheidet drei verschiedene Appraisal-Typen:

* **Class A Appraisal (SCAMPI A)**
  + einzige Bewertungsmethode, die zu einer Bewertung des Reifegrads oder des Fähigkeitsniveaus führt.
  + kann nur von einem zertifizierten SCAMPI SEU-autorisierten Lead Appraiser durchgeführt werden
    - hoher Anspruch an Zuverlässigkeit und Genauigkeit der Ergebnisse
    - hoher Aufwand, ca. einmal im Jahr
* **Class B Appraisal**
  + häufig als Benutzerakzeptanz- oder Testbewertung verwendet
    - mittlerer Anspruch an Zuverlässigkeit und Genauigkeit der Ergebnisse
    - mittlerer Aufwand, ca. dreimal im Jahr
* **Class C Appraisal**
  + gut geeignet für die laufende Fortschrittskontrolle bzw. die kontinuierliche Beobachtung der Prozessverbesserung
  + wird normalerweise als Tool zur Lückenanalyse und Datenerfassung verwendet
    - niedriger Anspruch an Zuverlässigkeit und Genauigkeit der Ergebnisse
    - niedriger Aufwand, häufig

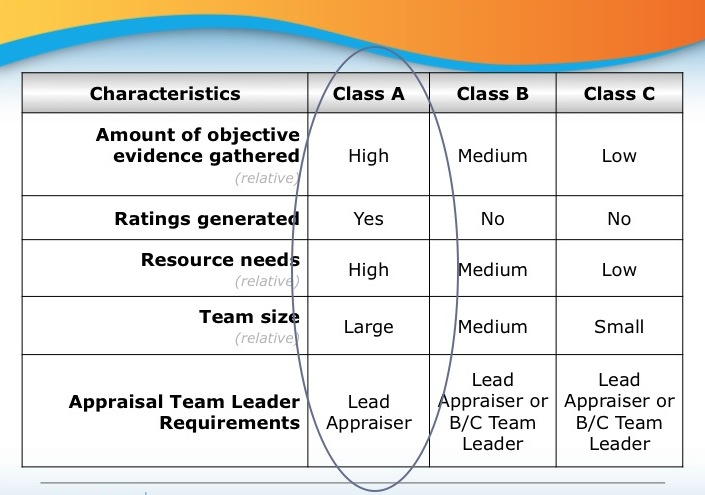


Abbildung ‑: SCAMPI Charakteristiken

Anhand der o.g. Appraisal-Methoden wird die Umsetzung des CMMI in einer Organisation geprüft. Ein Appraisal nach der SCAMPI-Vorgehensweise kann nur von einem SEI-autorisierten Lead Appraiser durchgeführt werden. Ein Appraisal mit einem Reifegrad ist in der Industrie als eine Auszeichnung anerkannt.

## 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:

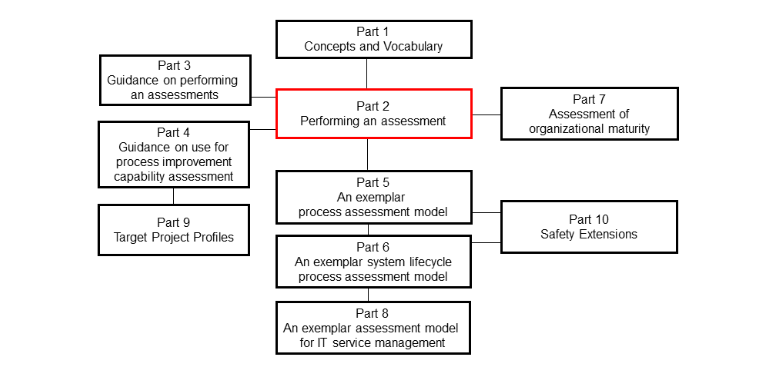


Abbildung ‑: Interdependenzen zwischen ISO 15504 Bänder

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.

Das SPICE Konzept teilt sich für die Prozessbeurteilung sowie -verbesserung in folgende wesentliche Bestandteile und Kriterien auf:

* 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. 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.

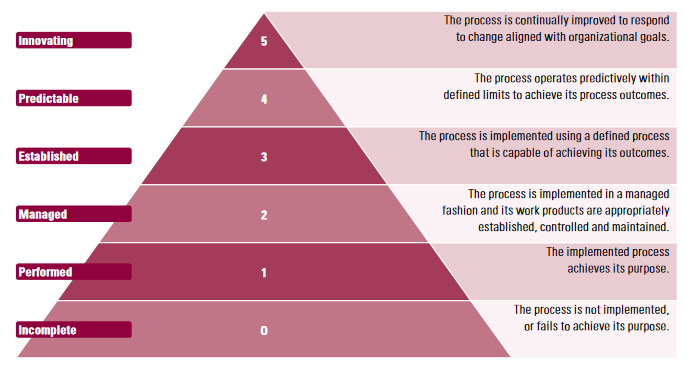


Abbildung ‑: SPICE Capability Levels

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.

# 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.

## Kritische Betrachtung

Ein agiler Entwicklungsprozess nach SCRUM zeichnet sich dadurch aus, dass in der Entwicklungsphase so früh wie möglich eine lauffähige Software (Increment) erstellt wird, die in regelmäßigen Abständen (Sprint-Reviews) dem Kunden (Product Owner) vorgezeigt werden kann. Dadurch ist es möglich flexibel auf weitere Wünsche einzugehen, um so die Kundenzufriedenheit zu erhöhen.

Im Hinblick auf unser Projekt SAPlexa ist es schwer klassische Vorgehensmodelle wie dem V-Modell oder dem RUP anzuwenden, da wir nach der agilen Methode SCRUM vorgegangen sind. Um eine kritische Betrachtung durchführen zu können, werden im Folgenden die einzelnen Projektentwicklungsschritte aus dem agilen SCRUM-Projekt in Anlehnung an die ISO 12207 herausgeschrieben.

Im ersten Schritt wurden zunächst die Anforderungen an das Projekt gesammelt und das Projektziel definiert. Dies wurde vom Product Owner im Product BackLog festgehalten. Anschließend wurden die jeweiligen Zuständigkeiten und Rollen innerhalb des SCRUM-Teams verteilt.

Nachdem die Anforderungen und das Projektziel festgehalten wurden, kam es zur Planung der Systemspezifikation. Darunter fallen die Recherche und Auswahl der Sprachsteuerung und die erste Modellierung einer GUI-Oberfläche.

Die nächste Prozessphase beschäftigt sich mit der Umsetzung der gewonnenen Informationen aus der Recherche. Das Ziel dabei war es eine lauffähige Sprachsteuerung zu programmieren, die das gesprochene Wort erkennt und in der GUI visualisiert. Um den Bezug zum Benutzer (hier: Lagerist) herzustellen, wurde die GUI aus der Nutzersicht entwickelt. Dadurch ist es möglich eine hohe Nutzerfreundlichkeit zu garantieren.

Da wir als agiles SCRUM-Team agiert haben, bewegten wir uns oftmals zwischen Testphase, Umsetzungsphase und Planungsphase. Der Grund dafür waren kontinuierliche Verbesserungsansätze aus kundenspezifischen Anforderungen, die in die Implementierung eingebaut werden sollten. Daraus wurden unsere Product Increments bei jeder Implementierung stabiler, qualitativer und näherten sich so immer mehr an das Projektziel.

**Bewertung**

Viele Faktoren, die in die Prozessbewertung mit einfließen, konnten in unserem Fallbeispiel nicht einbezogen werden, da die Entwicklung – entgegen der Entwicklung in einem Unternehmen – keinen kostenverursachenden Vorgang darstellte. Der Kostenfaktor kann als wesentlicher Bestandteil der Modelle daher nicht in der Betrachtung der Prozesse mitaufgenommen werden.

Hingegen standen Faktoren wie Qualitätsmanagement, Zeitmanagement und auch generelle Anforderungen an das Projekt im Mittelpunkt.

Im Hinblick auf das Qualitätsmanagement konnten viele Teilbereiche der Implementierung qualitativ verbessert und funktionell ausgeweitet werden. So konnte die zunächst fehlerbehaftete Sprachsteuerung eine höhere Wiedererkennung der Spracheingabe erlangen indem wir Anpassungsmethoden durchgeführt haben.

Für die konkrete Bewertung der Entwicklungsprozesse sollten konkrete Messwerte aus ausgewählten, zielorientierten Kriterien eine solide Prozessbeurteilung ermöglichen. Die Einstufungen im Rahmen des CMMI Bewertungsmodells werden als Appraisal, also Einschätzungen, vorgenommen. Daher soll im Fortgang der kritischen Betrachtung eine subjektive Einschätzung zur Festlegung der CMMI Fähigkeitsgrade helfen.

Aufgrund der agilen Entwicklungsmethodik erfolgte die Implementierung und Programmierung dynamisch, d.h. Prozesse wurden beobachtet, auf Ihren Output geprüft und ggf. in gewünschte Richtungen gelenkt und optimiert. Die Einschätzung unseres Entwicklungsprozesses übertrifft somit nach dem CMMI Modell die Fähigkeitsgrade 0 und 1 bei Weitem. Unsere Einschätzung liegt damit auf Höhe des Fähigkeitslevels 2. Die Standardisierung des Prozesses gemäß Fähigkeitslevel 3 erfolgte im Rahmen unserer Entwicklungsarbeit nicht.

Im Hinblick auf die Reifegradbeurteilung stellen wir fest, dass wir aufgrund der fehlenden Standardisierung der Prozesse den Reifegrad auf Level 3 nicht erreichen. Schätzungsweise bewegen wir uns somit zwischen Level 2 (Managed) und Level 3 (Defined).

**Fazit**

Die heutigen Prozessphasen einer Softwareentwicklung unterscheiden sich stark von früher. In der Vergangenheit werden Softwareentwicklungen klassisch nach vordefinierten Regeln für jede Prozessphase organisiert. Doch der immer größere Einfluss der Umwelt führte zu einem Wechsel der Art und Weise wie Entwicklungsprozesse durchgeführt werden sollen. Die Umwelt ist nicht vorhersehbar und ist im ständigen Wandel. Das führt dazu, dass der Softwareentwicklungsprozess zu jederzeit flexibel angepasst werden muss, um auf große Änderungen schnell reagieren zu können.

Diese ausschlaggebenden Faktoren wie die Flexibilität oder die schnelle Reaktion auf Veränderungen führten die agile Softwareentwicklung ein. Die Umwelt ist nun Teil des Softwareentwicklungsprozesses und wirkt sich zur Laufzeit auf alle Phasen aus. Neue Kundenanforderungen werden direkt in die Planung miteinbezogen. Schwachstellen werden durch regelmäßige Unit-Tests sofort erkannt und behoben. Projektziele können auf plötzliche Veränderungen leichter angepasst werden.

Die klassischen Standards und Vorgehensmethoden decken diese Problematik nur teils ab, da die klassischen Entwicklungsprozesse lange angelegt werden und dadurch erschwert auf geänderte Anforderungen reagiert werden können. Hingegen werden bei agilen Methoden die Anforderungen erst kurz vor Beginn der Umsetzung und Testvorbereitung ausformuliert. Dadurch stellen nachträgliche Änderungen keine Probleme für den Entwicklungsprozess dar.

Selbst aktuelle Studien der Hochschule Koblenz belegen, dass agile Methodiken eine weitaus höhere Wahrscheinlichkeit im Hinblick auf einen erfolgreichen Abschluss des Projekts aufweisen. Damit stellt die Anwendung (beispielsweise) der Wasserfallmethodik mit nur 14% einen vergleichsweisen schwachen Ansatz dar.

# Quellen- und Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | Abbildung 1.1, RUP-Modell. Online verfügbar unter: [<http://www.infforum.de/themen/anwendungsentwicklung/se-rup.htm>].  Letzter Zugriff: 14.02.2020 |
| [2] | Dietmar, Abts, Springer Vieweg, Grundkurs Wirtschaftsinformatik: Eine kompakte und praxisorientierte Einführung (2013), 8. Auflage. |
| [3] | Academic, 2000-2020. Online verfügbar unter: <https://deacademic.com/dic.nsf/dewiki/1161101>. Letzter Zugriff: 14.02.2020 |
| [4] | Wagner, Reinhard, Hrsg. Grau, Nina, Basiswissen Projektmanagement – Prozesse und Vorgehensmodelle (2013). Online verfügbar unter: [<https://www.projektivisten.de/fileadmin/user_upload/downloads/Basiswissen-Projektmanagement-Prozesse-und-Vorgehensmodelle.pdf#page=108>. Letzter Zugriff: 14.02.2020 |
| [5] | Abbildung 1.3 Wasserfallmodell. Online verfügbar unter: [<http://spolwig.de/is/softwareprojekte/slc_3.htm>]. Letzter Zugriff: 14.02.2020 |
| [6] | Dr. S. Streitz, 2. Saarbücker Standard zur Bewertung von Software, NUW-CoR, 4/95,270. <https://www.shp-itexperts.de/2-saarbruecker-standard-zur-bewertung-von-software>. Letzter Zugriff: 14.02.2020 |
| [7] | Russwurm S. (2013) Software: Die Zukunft der Industrie. In: Sendler U. (eds) Industrie 4.0. Xpert.press. Springer Vieweg, Berlin, Heidelberg, Online-ISBN: 978-3-642-36917-9 |
| [8] | Glinz, Martin, Informatique, 6/1999 [<https://files.ifi.uzh.ch/rerg/arvo/ftp/papers/INF_6_99.pdf>]. Letzter Zugriff: 14.02.2020 |
| [9] | Schmied, Jürgen, Wentzel, Paul-Roux, Gerdom, Michael, Uwe Hehn: Mit CMMI Prozesse verbessern! dpunkt.verlag, 2008, [ISBN 978-3-89864-538-6](https://de.wikipedia.org/wiki/Spezial:ISBN-Suche/9783898645386). |
| [10] | Wallmüller, Ernest, SPI - Software Process Improvement mit CMMI und ISO 15504. Hanser, München 2006, [ISBN 3-446-40492-9](https://de.wikipedia.org/wiki/Spezial:ISBN-Suche/3446404929). |
| [11] | Andreas Bärwald (2018) Sichere Werkzeugketten und werkzeugunterstütztes Assessment. Online verfügbar unter: <https://epub.uni-regensburg.de/38145/1/Dissertation_Andreas_Baerwald_2018-12-17_final.pdf>. Letzter Zugriff: 14.02.2020 |
| [12] | Dumke, R., Ebert, C., Heidrich, J. et al. Informatik Spektrum (2013) 36: 508. <https://doi.org/10.1007/s00287-013-0723-y>, Springer-Verlag Berlin Heidelberg 2013, Online-ISBN: 1432-122X. Letzter Zugriff: 14.02.2020 |
| [13] | Mohsen R. (2004) Prozess- und Technologiemanagement in der Softwareentwicklung, Oldenbourg Wissenschaftsverlag GmbH, München, Wien, ISBN: 3-486-27549-6 |
| [14] | Die Beauftragte der Bundesregierung für Informationstechnik. V-Modell XT (Portal des Standards). Online verfügbar unter: [www.v-modell-xt.de](http://www.v-modell-xt.de). Letzter Zugriff: 14.02.2020 |
| [15] | Sprunck, Markus, ISO/IEC 15504 (SPICE), 2007 [<http://www.se-konferenzen.de/bisher/se2007/folien/se2007-B2-3.pdf>]. Letzter Zugriff: 14.02.2020 |
| [16] | Maag, Kugler, Automotive Spice v.3.1 POCKET GUIDE, 2018 <https://www.kuglermaag.de/fileadmin/05_CONTENT_PDF/2-10_automotive-spice_version_3_pocketguide.pdf>. Letzter Zugriff: 14.02.2020 |
| [17] | Jutta Eckstein: *Agile Softwareentwicklung im Großen. Ein Eintauchen in die Untiefen erfolgreicher Projekte.* Deutsche Bearbeitung von Nicolai Josuttis. dpunkt-Verlag, Heidelberg 2004, [ISBN 3-89864-250-X](https://de.wikipedia.org/wiki/Spezial:ISBN-Suche/389864250X). |
| [18] | Henning Wolf, Wolf-Gideon Bleek: *Agile Softwareentwicklung. Werte, Konzepte und Methoden.* 2. aktualisierte und erweiterte Auflage. dpunkt-Verlag, Heidelberg 2010, [ISBN 978-3-89864-701-4](https://de.wikipedia.org/wiki/Spezial:ISBN-Suche/9783898647014). |
| [19] | [*Status Quo Agile*](http://www.status-quo-agile.de/) auf status-quo-agile.de, Studie zu Verbreitung und Erfolgen agiler Methoden. Letzter Zugriff: 14.02.2020 |
| [20] | Andreas Nerlich, SCRUM Einleitung, Präsentationsdokument, Online verfügbar auf: [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. Letzter Zugriff: 24.02.2020 |
|  |  |
| [21] | Mario Janschitz, T3N, Was ist SCRUM?, Online verfügbar auf: <https://t3n.de/news/scrum-was-ist-das-506705/>  Letzter Zugriff: 14.02.2020 |
| [22] | Alexander Kriegisch, SCRUM Master, Online verfügbar auf: <https://scrum-master.de/>  Letzter Zugriff. 14.02.2020. |

**Arbeitseinteilung im Projekt:**

|  |  |
| --- | --- |
| **Autor** | **Zuständigkeitsbereich** |
|  |  |
| Angelo Cavallaro | Kapitel 4.1, 4.2, 4.3, 5.1, 5.2, 5.3 |
| Beyza Gürler | Kapitel 1.1, 1.3, 2, 3.1, 5.1, 5.2, 5.3, Abstract |
| David Süßlin | Kapitel 1.1, 3.1, 3.2, 3.3, 5.1, 5.2, 5.3 |
| Hüseyin Kasarca | Kapitel 1.2, 4.1, 4.2, 5.1, 5.2, 5.3 |
| Nahom Ghebremussie | Kapitel 1.4, 4.1, 4.2, 5.1, 5.2, 5.3, Abstract |
| Viktor Sellmann | Kapitel 1.1, 3.1, 3.2, 3.3, 5.1, 5.2, 5.3 |

1. Andreas Nerlich, SCRUM Einleitung. [↑](#footnote-ref-1)
2. Mario Janschitz, T3N [↑](#footnote-ref-2)
3. Mario Janschitz, T3N [↑](#footnote-ref-3)
4. Mario Janschitz, T3N [↑](#footnote-ref-4)
5. Alexander Kriegisch, SCRUM Master [↑](#footnote-ref-5)
6. Alexander Kriegisch, SCRUM Master [↑](#footnote-ref-6)