Prozessmodelle / Vorgehensmodelle

Weiterführende Literatur:

- Schatten, *Best Practice Software-Engineering*, Seite 47-69

Ein Prozessmodell ist ein für die Softwareentwicklung angepasstes Vorgehensmodell bei der professionellen Anwendungsentwicklung. Es dient dazu, die Softwareentwicklung übersichtlicher zu gestalten und in der Komplexität beherrschbar zu machen.[1]

Beispiele

- Wasserfallmodell
- V-Modell
- Spiralmodell
- Agile Entwicklung

Wasserfallmodell

Weiterführende Literatur:

- Schatten, *Best Practice Software-Engineering*, Kapitel 3.2 „Wasserfallmodell“, Seite 48-49
- *Softwaresysteme: Präsenztag 1: Foliensatz: Lebenszyklus, Vorgehensmodelle, Projektmanagement*, Seite 26
- Schneider, *Taschenbuch der Informatik*, Kapitel 8.2.3 Plangetriebene Vorgehensmodelle, Seite 226-227
- *Wikipedia-Artikel „Wasserfallmodell“*

Das Wasserfallmodell ist der Klassiker unter den Vorgehensmodellen. Das Modell wurde bereits in den 1970er Jahren veröffentlicht und wird heute nur noch selten für bestimmte Anwendungsgebiete eingesetzt, bei denen ein *strikt sequenzieller Projektablauf* möglich ist. Namensgebend ist, dass

strikt sequenzieller Projektablauf

[1] *Softwaresysteme: Präsenztag 1: Foliensatz: Lebenszyklus, Vorgehensmodelle, Projektmanagement*, Seite 25.

die sequenziellen Schritte des Lebenszyklusses eines Software-Produkts in Form eines Wasserfalls dargestellt werden.

Alle Schritte werden sequenziell durchlaufen, wobei jeweils die *vorangegangene Phase vollständig abgeschlossen* und freigegeben werden muss, *bevor die nächste Phase gestartet* werden kann. Als *Abschluss jeder Phase* sind Verifikations- und *Validierungsschritte* vorgesehen, die quasi als Freigabe der erstellten Produkte verwendet werden. Wird die geforderte Qualität an der Stelle nicht erreicht, kann zur vorangegangenen Phase *„zurückgesprungen“* werden.[2]

vorangegangene Phase vollständig abgeschlossen

bevor die nächste Phase gestartet

Validierungsschritte

„zurückgesprungen“

Aufgrund der *guten Planbarkeit* durch klar abgegrenzte Phasen wird das Wasserfallmodell typischerweise bei stabilen und *genau beschriebenen Anforderungen* und Abläufen eingesetzt.

genau beschriebenen Anforderungen

Nachteilig wirkt sich allerdings aus, dass dieses Modell *nur wenig Flexibilität* aufweist und somit nur schwer auf (späte) Änderungen im Projekt reagieren kann. Als weitere Folge werden Fehler unter Umständen erst sehr spät im Entwicklungsprozess oder erst beim Kunden erkannt. Das hat zur Folge, dass teuere Nacharbeiten notwendig werden und das Projekt in Zeitnot kommen kann.

Ein weiterer Nachteil des Wasserfallmodells ist, dass durch den sequenziellen Ablauf eine *finale Software-Lösung erst sehr spät zur Verfügung* steht, der Kunde also erst gegen Ende des Projekts sieht, was er tatsächlich bekommen wird und dann grundlegende Änderungen nicht mehr leicht machbar sind.

finale Software-Lösung erst sehr spät zur Verfügung

In der Praxis der modernen Software-Entwicklung ist das Modell gelegentlich noch anzutreffen, wird aber zunehmend *von flexibleren Prozessmodellen verdrängt.*

von flexibleren Prozessmodellen verdrängt.

Phasen[5]

(a) Anforderungsanalyse und -spezifikation (Requirement analysis and specification) resultiert im Lastenheft

(b) Systemdesign und -spezifikation (System design and specification) resultiert in der Softwarearchitektur

(c) Programmierung und Modultests (Coding and module testing) resultiert in der eigentlichen Software

(d) Integrations- und Systemtest (Integration and system testing)

[2] Schatten, *Best Practice Software-Engineering*, Seite 48.

[5] *Wikipedia-Artikel „Wasserfallmodell“.*

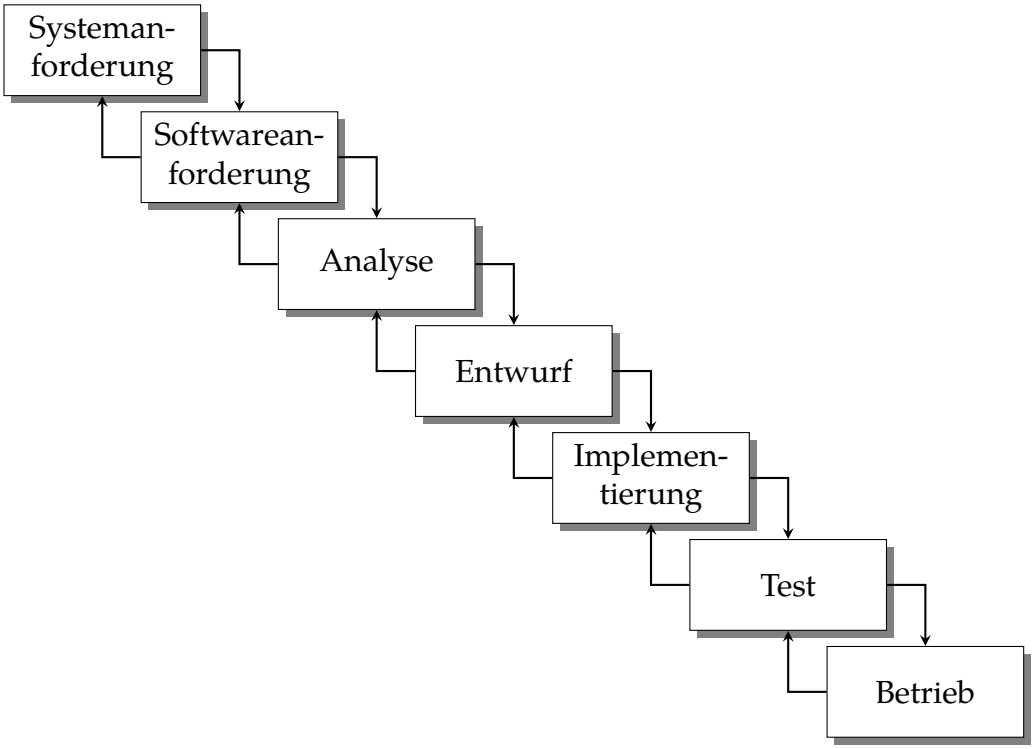


Abbildung 1.1: Nach researchgate.net[3]

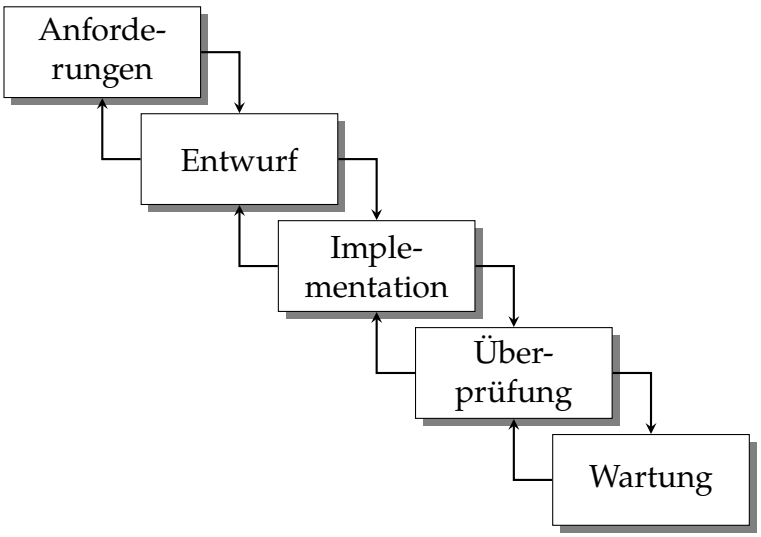


Abbildung 1.2: Nach Wikipedia[4]

(e) Auslieferung, Einsatz und Wartung (Delivery, deployment and maintenance)

Eine andere Variante macht daraus sechs Schritte:

(a) Planung (mit Erstellung des Lastenhefts, Projektkalkulation und Projektplan) (Systems Engineering)

(b) Definition (mit Erstellung des Pflichtenhefts, Produktmodell, GUI-Modell und evtl. schon Benutzerhandbuch) (Analysis)

(c) Entwurf (UML, Struktogramme) (Design)

(d) Implementierung (Coding)

(e) Testen (Testing)

(f) Einsatz und Wartung (Maintenance)

V-Modell

Weiterführende Literatur:

- Schatten, *Best Practice Software-Engineering*, Seite 49-52
- *Softwaresysteme: Präsenztag 1: Foliensatz: Lebenszyklus, Vorgehensmodelle, Projektmanagement*, Seite 27
- Schneider, *Taschenbuch der Informatik*, Kapitel 8.2.3 Plangetriebene Vorgehensmodelle, Seite 226-227
- *Wikipedia-Artikel „V-Modell“*

Das V-Modell organisiert *ähnlich dem Wasserfallmodell* den Softwareentwicklungsprozess in *Phasen*. *Zusätzlich* zu diesen Entwicklungsphasen definiert das V-Modell auch das Vorgehen zur *Qualitätssicherung* (*Testen*), indem den einzelnen *Entwicklungsphasen Testphasen gegenüber gestellt* werden. Auf der linken Seite wird mit einer funktionalen/fachlichen Spezifikation begonnen, die immer tiefer detailliert zu einer technischen Spezifikation und Implementierungsgrundlage ausgebaut wird. In der Spitze erfolgt die Implementierung, die anschließend auf der rechten Seite gegen die entsprechenden Spezifikationen der linken Seite getestet wird. So entsteht bildlich das namensgebende „V“, welches die einzelnen Entwicklungsebenen ihren jeweiligen Testebenen gegenüberstellt.[6]

ähnlich dem Wasserfallmodell

Phasen

Zusätzlich

Qualitätssicherung (Testen)

Entwicklungsphasen Testphasen gegenüber gestellt

[6] *Wikipedia-Artikel „V-Modell“*.

Erweiterung des Wasserfall-Modells im Hinblick auf die beiden Qualitätssicherungsaspekte

Verifikation Überprüfung der Übereinstimmung zwischen einem Software-Produkt und der Spezifikation

Frage: Are we doing things right?

Validierung Überprüfung der Eignung eines Softwareprodukts bezogen auf seinen Einsatzzweck

Frage: Are we doing right things?[7]

Inkrementelles Vorgehen[8]

Eine inkrementelle – also schrittweise – Vorgehensweise wird vor allem bei großen und *komplexen Systemen* eingesetzt, von denen rasch erste verwendbare Teile ausgeliefert werden sollen. Ein Hauptziel ist dabei, möglichst *rasch* mit einer *(Teil-)Lösung* auf den Markt zu kommen und dann das Software-Produkt *durch laufende Ergänzungen auszubauen*.

komplexen Systemen

rasch

(Teil-)Lösung

durch laufende Ergänzungen auszubauen

Die Eckpunkte der Entwicklung sind: *Analyse*, *Design*, *Implementierung*, *Integration* und *Auslieferung*. Dabei werden *alle Phasen für eine Version* (oder ein Release) durchlaufen, bis diese Version abgeschlossen ist und ausgeliefert werden kann. *Parallel bzw. zeitlich versetzt* kann bereits mit der Planung und Umsetzung der *Folgeversion* begonnen werden.

Analyse

Design

Implementierung

Integration

Auslieferung

alle Phasen für eine Version

Parallel bzw. zeitlich versetzt

Folgeversion

Spiralmodell

Weiterführende Literatur:

- Schatten, *Best Practice Software-Engineering*, Seite 57-58
- *Softwaresysteme: Präsenztag 1: Foliensatz: Lebenszyklus, Vorgehensmodelle, Projektmanagement*, Seite 28
- *Wikipedia-Artikel „Spiralmodell“*

Das Spiralmodell stellt eine *konkrete Ausprägung der inkrementellen Entwicklung* dar, in dem *vier grundlegende Schritte* solange zyklisch durchlaufen werden, bis das Produkt in einer zufriedenstellenden Qualität vorliegt

konkrete Ausprägung der inkrementellen Entwicklung

vier grundlegende Schritte

[7] *Softwaresysteme: Präsenztag 1: Foliensatz: Lebenszyklus, Vorgehensmodelle, Projektmanagement*, Seite 27.

[8] Schatten, *Best Practice Software-Engineering*, Seite 56.

(a) Definition von Zielen und Alternativen.

(b) Einschätzung des Risikos.

(c) Entwicklung und Durchführung von Tests und Evaluierungen der aktuellen Ergebnisse.

(d) Feedback zur erstellten Lösung und Planung für die nächste Iteration.[9]

Agile Vorgehensmodelle

Weiterführende Literatur:

- Schatten, *Best Practice Software-Engineering*, Seite 62-65
- *Softwaresysteme: Präsenztag 1: Foliensatz: Lebenszyklus, Vorgehensmodelle, Projektmanagement*, Seite 29-31
- Schneider, *Taschenbuch der Informatik*, Kapitel 8.2.4 Agile Modelle, Seite 229-231

Kritikpunkte an traditionellen Vorgehensmodellen

Kritikpunkte an traditionellen Vorgehensmodellen sind unter anderem mangelhafte Flexibilität (durch vorgegebene starre Strukturen), mangelnde Einbeziehung des Kunden in den Entwicklungsprozess sowie ein (scheinbar unnötig) hoher Dokumentationsaufwand. Agile Software-Entwicklungsprozesse sollen diese Nachteile beheben und ein *höheres Maß an Flexibilität und Kundennähe* bei einem *Mindestmaß an Dokumentation* ermöglichen. Agile Entwicklungsprozesse gehen auf das *Agile Manifest*, das von 17 Software-Entwicklern im Jahr 2001 veröffentlicht wurde, zurück:

höheres Maß an Flexibilität und Kundennähe

Mindestmaß an Dokumentation

Agile Manifest

> *„Individuals and interactions over processes and tools*
> *Working software over comprehensive documentation*
> *Customer collaboration over contract negotiation*
> *Responding to change over following a plan.“*

SCRUM

SCRUM definiert ein agiles Software-Projekt aus der Sicht des Projektmanagements und besteht aus einer Sammlung von Prozeduren, Rollen und Methoden zur erfolgreichen Projektdurchführung. Das Prozessmodell umfasst drei wesentliche Phasen:

Pregame In einem *Pregame* erfolgt die Festlegung der wesentlichen Pro- Pregame

[9] Schatten, *Best Practice Software-Engineering*, Seite 57.

dukteigenschaften und der grundlegenden Architektur sowie die Projektplanung. Alle Eigenschaften und gewünschten Features werden *gemeinsam mit dem Kunden* in einem Produkt-Backlog gesammelt und priorisiert. In diesem *Produkt-Backlog* werden auch geänderte Anforderungen gesammelt.

gemeinsam mit dem Kunden

Produkt-Backlog

Sprint Die eigentliche Entwicklungsarbeit wird in einem *Sprint* durchgeführt. Vor einem Sprint werden die *wichtigsten und machbaren Features* aus dem priorisierten Produkt-Backlog ausgewählt und in das Sprint-Backlog übernommen. Wichtig ist dabei, dass die Auswahl und Planung nur so viele Features umfasst, wie das Team im nächsten Sprint auch tatsächlich umsetzen kann. Das *Sprint-Backlog* beinhaltet also einen *machbaren Auszug aus dem Produkt-Backlog*.

Sprint

wichtigsten und machbaren Features

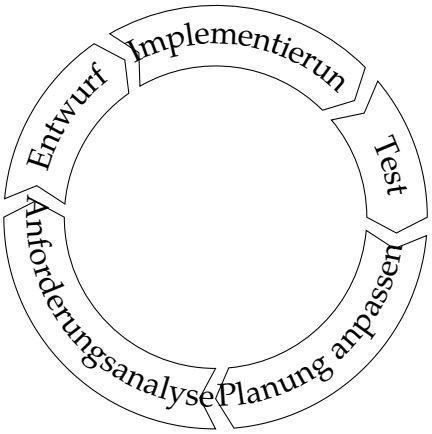
Sprint-Backlog

machbaren Auszug aus dem Produkt-Backlog

Postgame Die abschließende Phase – das *Postgame* – umfasst die *Bereitstellung und Auslieferung neuer Funktionalität* in Form von (neuen) Releases.[10]

Postgame

Bereitstellung und Auslieferung neuer Funktionalität



Literatur

[1] Alexander Schatten. *Best Practice Software-Engineering. Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen*. 2010.

[2] Uwe Schneider. *Taschenbuch der Informatik*. 7. Aufl. Hanser, 2012. ISBN: 9783446426382.

[10] Schatten, *Best Practice Software-Engineering*, Seite 63.

[3] *Softwaresysteme: Präsenztag 1: Foliensatz: Lebenszyklus, Vorgehensmodelle, Projektmanagement*. https://www.studon.fau.de/file2703521_download.html.

[4] *Wikipedia-Artikel „Spiralmodell“*. https://de.wikipedia.org/wiki/Spiralmodell.

[5] *Wikipedia-Artikel „V-Modell“*. https://de.wikipedia.org/wiki/V-Modell.

[6] *Wikipedia-Artikel „Wasserfallmodell“*. https://de.wikipedia.org/wiki/Wasserfallmodell.