12 Grundlagen der Softwareentwicklung

- Wie - konzeptionelle Vorgehensweise beim Entwurf von Software
- Was - Phasenmodell des Software-Lebenszyklus
- Welche Vorgehensmodelle
- wie kann man Software testen
- wie findet man Fehler in der Software

12.1 Software entwickeln

- Abhängig von der zu erstellenden Software werden bei der professionellen Programmierung Vorgehensmodelle gewählt, die festlegen, wie die Software entwickelt wird
- eine Vorgehensmodell beschreibt, welche Prinzipien, Methoden und Darstellungsmittel eingesetzt werden
- Vorgehensmodelle sind in großen Projekten notwendig, um die Entwicklung der Software in verwaltbare und kontrollierbare Teile zu gliedern
- Prinzip der Modularität:
    - eine Gesamtaufgabe wird in Teilaufgaben (Module) zerlegt
        - reduziert die Komplexität und
        - führt zu besserer Verständlichkeit
    - Module können einzeln programmiert werden
    - Module können in verschiedenen Programmen verwendet werden
    - Module besitzen Schnittstellen als Verbindungsstellen zwischen den Modulen selbst oder zwischen dem Programm und dem Anwender
- Prinzip der Abstraktion:
    - Auswahl relevanter Informationen aus einer größeren Menge von verfügbaren Informationen
    - unwesentliche Informationen von der Aufgabenstellung ausschließen

12.2 Methoden

- Top-Down-Methode
  - Ausgangspunkt ist die Gesamtaufgabe
  - sie wird in Teilaufgaben zerlegt
  - bei komplexen Aufgabenstellungen können die Teilaufgaben in weitere Teilaufgaben aufgeteilt werden
- Bottom-up-Methode
  - empfohlene Vorgehensweise, falls die Aufgabenstellung nicht exakt beschrieben ist
  - einzelne Module werden zu einem großen Modul zusammengesetzt
  - beim Zusammensetzen der Module kann es zu Problemen kommen, da das Gesamtsystem nicht genau festgelegt war
- Up-Down-Methode (Gegenstromverfahren)
  - bei dieser Strukturierungsmethode wird die Gesamtaufgabe durch die Top-Down-Methode verfeinert und es werden Teilaufgaben bottom-up abstrahiert
  - so lassen sich kritische Teilaufgaben zuerst testen

12.3 Software-Lebenszyklus

1. Analyse

Aufgabe	**Was** soll die Software tun? Definition der Anforderungen • Funktionsumfang und Qualitätsmerkmale • Art der Benutzeroberfläche • Schnittstellen der Systemumgebung • eingesetzte Hardware während der Programmierung • Entwicklungssoftware (IDE = integrierte Entwicklungsumgebung) • Umfang der Dokumentation • Ein-/Ausgabedaten und Testdaten festlegen • Wirtschaftlichkeitsberechnungen
Test	Anforderungstest
Ergebnis	Pflichtenheft

2. Entwurf (Design)

Aufgabe	**Wie** ist die Software zu realisieren? • Strukturierter Softwareentwurf (z.B. Top-down-Methode) • Beschreibung der Algorithmen • Festlegung der Programmiersprache für die Implementierung • Festlegung der Programmierstukturen • Anlegen der Dokumentation
Test	• Entwurfstest der Dokumente und des Designs • Funktionstest des Prototyps
Ergebnis	Beschreibung des Entwurfs bzw. Softwarespezifikation

3. Implementierung

Aufgabe	• Programmcode in der festgelegten Programmiersprache erstellen • Programmcode testen und dokumentieren
Test	Funktions- und Modultest
Ergebnis	• Programmcode • Dokumentation zum Programmcode • Dokumentation zum Test des Programmcodes

4. Integration

Aufgabe	• Zusammenfügen der Einzelaufgaben zur Gesamtaufgabe • Dokumentation
Test	• Integrationstest testet das Zusammenspiel der Einzelteile

	• Systemtest soll gewährleisten, das die Anforderungen des Pflichtenhefts erfüllt werden
Ergebnis	komplettes System und Dokumentation für den Benutzer

5. Einsatz (Installation)

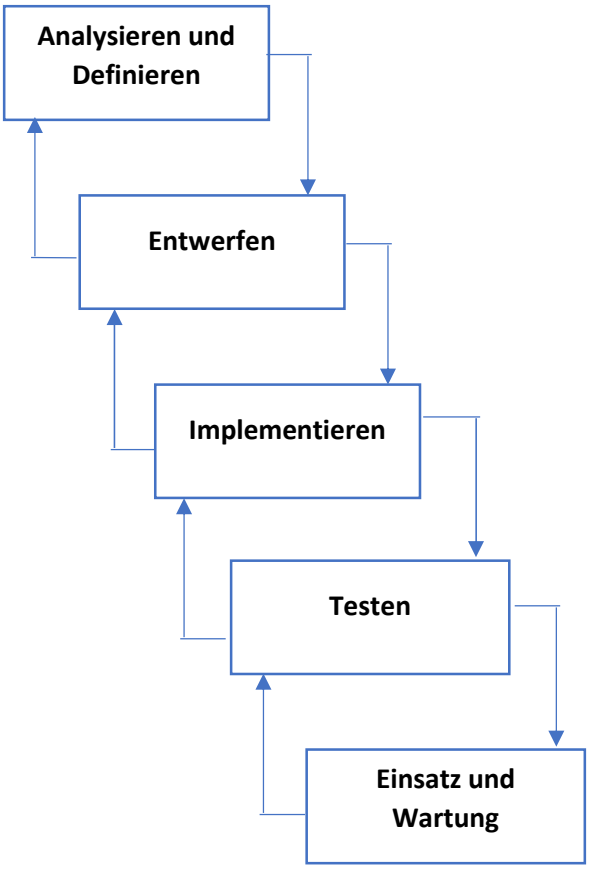
Aufgabe	• Software für die Auslieferung fertigstellen • Dokumentation fertigstellen
Test	Akzeptanztest oder Abnahmetest
Ergebnis	Software im Einsatz

6. Wartung

Aufgabe	• Korrektur auftretender Fehler • Anpassung der Systemumgebung • Änderungen und Erweiterungen
Test	Konfigurationstest – zum Prüfen der Sotwareanpassungen
Ergebnis	Änderungen und Erweiterungen der Funktionen

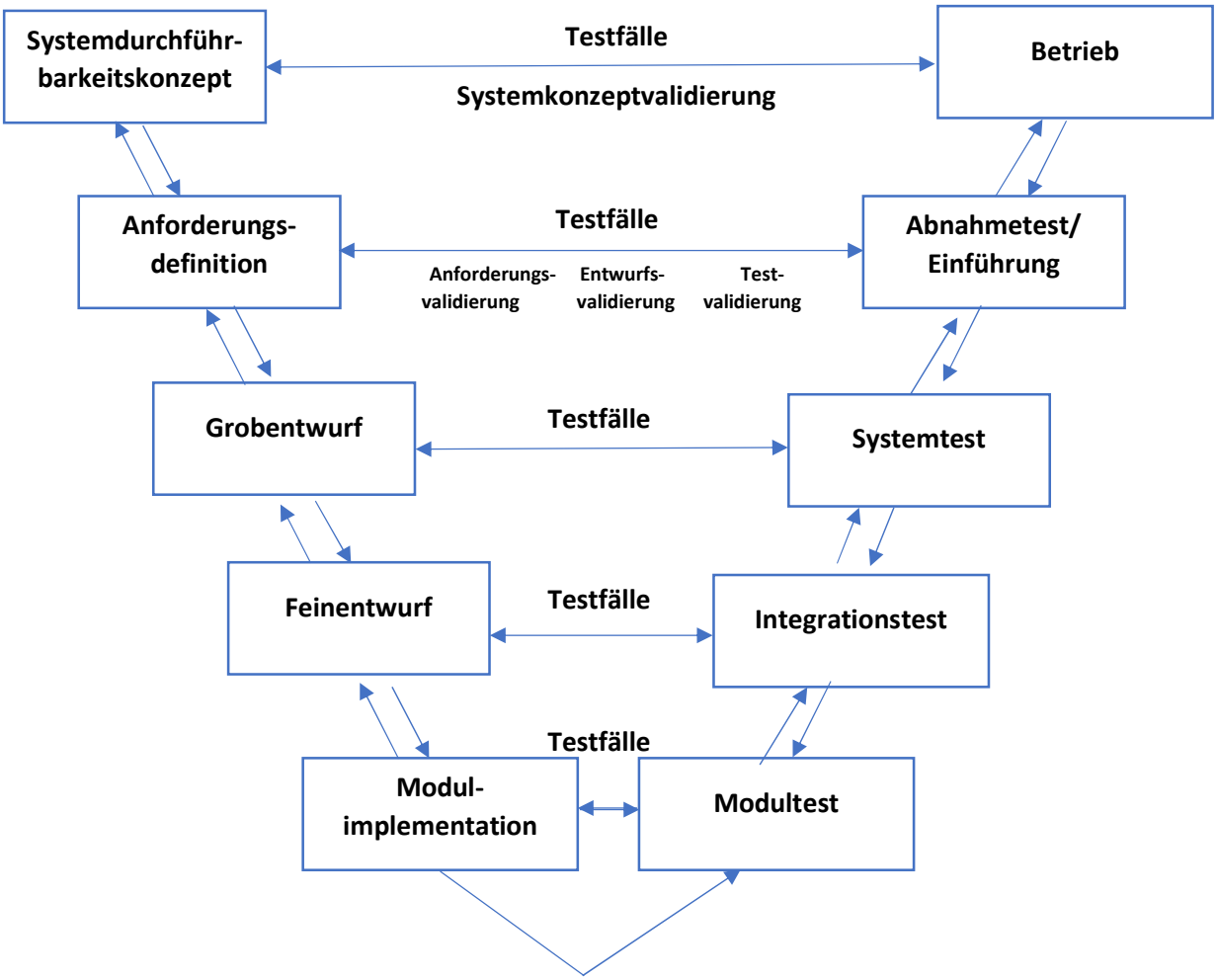
12.4 Vorgehensmodelle

12.4.1 Wasserfallmodell



- Vorteil:
  - geringer Managementaufwand
  - leicht verständlich
- Nachteil:
  - späte Änderungen sind nur mit großem Aufwand realisierbar
  - sehr unflexibel durch die strenge Sequenz

12.4.2 V-Modell



- Vorteil:
  - Gesamtmodell
  - gut geeignet für große Projekte
  - Qualitätssicherung steht im Vordergrund
  - kann angepasst und erweitert werden
- Nachteil:
  - Vorgehensweisen sind sehr allgemein
  - für kleine Projekt ungeeignet
  - sehr aufwendige Verwaltung
  - CASE-Werkzeuge für die Entwicklung erforderlich
    - Computer Aided Software Engineering = computergestützte Softwareentwicklung
    - Ziel:
      Produktivität und Qualität zu verbessern und das Management zu unterstützen
    - z. B.
      - Eclipse
      - VisualStudio
      - IDLE (Integrated Development and Learning Environment)
      - Modellierungsprogramme für die Diagrammerstellung (UML)

12.4.3 Scrum?

- Was ist Scrum?
  - ein Modell oder eine Methode des agilen Projektmanagement
- Was bedeutet Scrum?
  - alle Beteiligten eines Projekts treffen sich jeden Tag, um über anstehende Aufgaben und den bisherigen Verlauf im Projekt zu sprechen.
  - es gibt Regeln und Methoden, die bei der Zielerreichung helfen
  - das soll helfen, die flexibler und agiler zu werden
- Warum ist man mit Scrum agil?
  - In einem Scrum-Projekt gelten wenige und einfache Regeln.
  - Sie sind nur dafür da, das Projektteam zu unterstützen, um das gemeinsame Projektziel zu erreichen.
  - Die wichtigste Regel ist, dass ein Team beim agilen Projektmanagement sich selbst organisieren kann und darf.
  - Das Projektteam bekommt einen Auftrag und legt mit dem Auftraggeber die Projektziele fest.
- Methoden und Regeln beim Projekt-Sprint
  - Das Projekt läuft in mehreren Phasen ab.
  - Eine Phase wird als Projekt-Sprint bezeichnet.
  - In jedem Sprint gibt es die täglichen Meetings, das Daily Scrum.
  - Es gibt verschiedene Methoden und Tools, die bei der Prüfung und Durchführung der einzelnen Aufgaben und Aktivitäten des Sprints helfen.
  - Mit ihnen wird auch geprüft, was nach einem Sprint erreicht wurde.
  - Besonders bekannt: das Product Backlog, das Sprint Backlog, das Kanban-Board, das Sprint Review und das Burndown-Chart.
  - Immer wieder wird die Zusammenarbeit im Team reflektiert – die Sprint Retrospective.

12.4.3.1 Scrum-Rollen

- Welche Rollen gibt es bei Scrum-Projekten?
  - der Produkteigner (Product Owner)
  - der Scrum-Master
  - das Mitglied im Projektteam
- Produkteigner
  - Der Produkteigner vertritt die Anwender des Produkts oder die Stakeholder des Projekts. Das sind alle, die betroffen sind und ein Interesse am Erfolg des Projekts haben. Wenn eine neue Software oder IT-Lösung im Unternehmen eingeführt wird, sind das die Nutzer, denn sie wollen reibungslos mit ihren Programmen arbeiten. Bei Produkten sind es die Produktmanager, die als Stimme der Kunden auftreten. Der Produkteigner sollte wissen, was die Kunden haben wollen. Aber auch das Marketing, der Vertrieb und der Kundendienst können Anforderungen an das Projektteam stellen.
- Scrum-Master (Project-Master)
  - Der Scrum-Master trägt die Verantwortung für den Scrum-Prozess. Der Scrum-Master ist Moderator und Unterstützer für das Projektteam. Er beseitigt Hindernisse und fördert die gute Zusammenarbeit im Team. Er beschafft die notwendigen Ressourcen und ist Ansprechpartner für Außenstehende. Und er hilft dem Team bei methodischen Problemen und stellt sicher, dass die Regeln des agilen Projektmanagements eingehalten werden.

- Projektteam
    - Ein Scrum-Team sollte zwischen fünf und zehn Mitarbeiterinnen und Mitarbeitern umfassen. Die Teammitglieder organisieren alle Aufgaben selbst. Es gibt im Team keine Hierarchie. Jeder hat dieselben Rechte und Pflichten, aber unterschiedliche Kompetenzen. Alle Fachbereiche, die zur Lösung beitragen, sollten vertreten sein. Wichtig ist, dass alle Teammitglieder aus eigenem Antrieb dabei sind. Sie sollten sich ihre Projekte selbst aussuchen können. Das setzt Vertrauen des Managements und Verantwortungsbewusstsein der Mitarbeitenden voraus.

12.4.3.2 Die 5 Schritte des Scrum-Prozesses

- Am Anfang steht eine Produkt-Vision; eine Idee des Produkts, die der Auftraggeber des Projekts vorantreiben will und die auch den Anwendern einen Nutzen bringt. Eine solche grobe Vorstellung vom Produkt oder der Lösung, die im Scrum-Projekt erarbeitet werden soll, ist der Auftrag. Die Produkt-Vision wird in Story Cards oder User Stories überführt, die aus Sicht des Anwenders einzelne Elemente, Merkmale und Funktionen des Produkts beschreiben.
- Mit dieser Grundlage startet ein agiles Projekt nach Scrum. Dann umfasst der Scrum-Prozess folgende Schritte:

12.4.3.3 Product Backlog anlegen und pflegen

- Aus den Anforderungen des Produkteigners und den Story Cards wird ein sogenanntes Product Backlog zusammengestellt. Das ist eine Sammlung sämtlicher Funktionen und Merkmale, die das Produkt haben soll. Am Anfang ist diese Zusammenstellung noch grob, doch im Projektverlauf wird sie immer genauer.
- Im Product Backlog werden Prioritäten vergeben. Eine hohe Priorität erhalten die Elemente und Funktionen die am wichtigsten sind und eine hohe Zufriedenheit der Anwender sicherstellen. Andere Anforderungen sind nicht so wichtig, können aussortiert werden, werden mit anderen zusammengelegt, sind technisch nicht realisierbar oder werden verschoben. Sie werden dann bei der Überarbeitung oder bei der Erweiterung des Produkts behandelt.

12.4.3.4 Im Sprint Planning das Sprint Backlog erstellen

- Agiles Projektmanagement mit Scrum ist ein iterativer Prozess, der sich aus mehreren Sprints zusammensetzt, bis das gewünschte Produkt fertiggestellt ist. Der Sprint ist der Kern eines Scrum-Projekts. Er ist eine fest vorgegebene Zeitdauer (Time Box) von maximal einem Monat.
- Im Sprint Planning werden die Aufgaben für den als Nächstes anstehenden Sprint geplant und das Sprint-Ziel formuliert. Sie ergeben sich aus den Einträgen auf dem Product Backlog und den Prioritäten. Mit dem Sprint Planning wird geklärt:
- Was wird im nächsten Sprint entwickelt, erstellt oder durchgeführt?
- Wie werden die entsprechenden Aufgaben und Arbeiten erledigt?
- Das Scrum-Team erstellt daraus ein Sprint Backlog. Das ist eine Auswahl der Product-Backlog-Einträge für den nächsten Sprint ergänzt um den Umsetzungsplan. Außerdem ist klar, woran sichtbar ist, ob die Aufgabe erledigt und das Teil-Produkt (Inkrement) des Sprints erstellt ist. Die sogenannte „Definition of Done“ wird formuliert.
- Eine einzelne Aufgabe, die dann zu erledigen ist, wird Ticket genannt. Alle Tickets sind im sogenannten Sprint Backlog aufgeführt. Das ist gewissermaßen der Maßnahmenplan und der Arbeitsvorrat für das Entwickler-Team für den nächsten Sprint. Jedes Teammitglied übernimmt eigenverantwortlich einzelne Tickets (Verpflichtungserklärung). Im Sprint

arbeiten die Teammitglieder an ihren Aufgaben, den Tickets, bis diese fertig sind und das „Done“ erreicht ist.

12.4.3.5 Im Daily Scrum den Arbeitsfortschritt besprechen

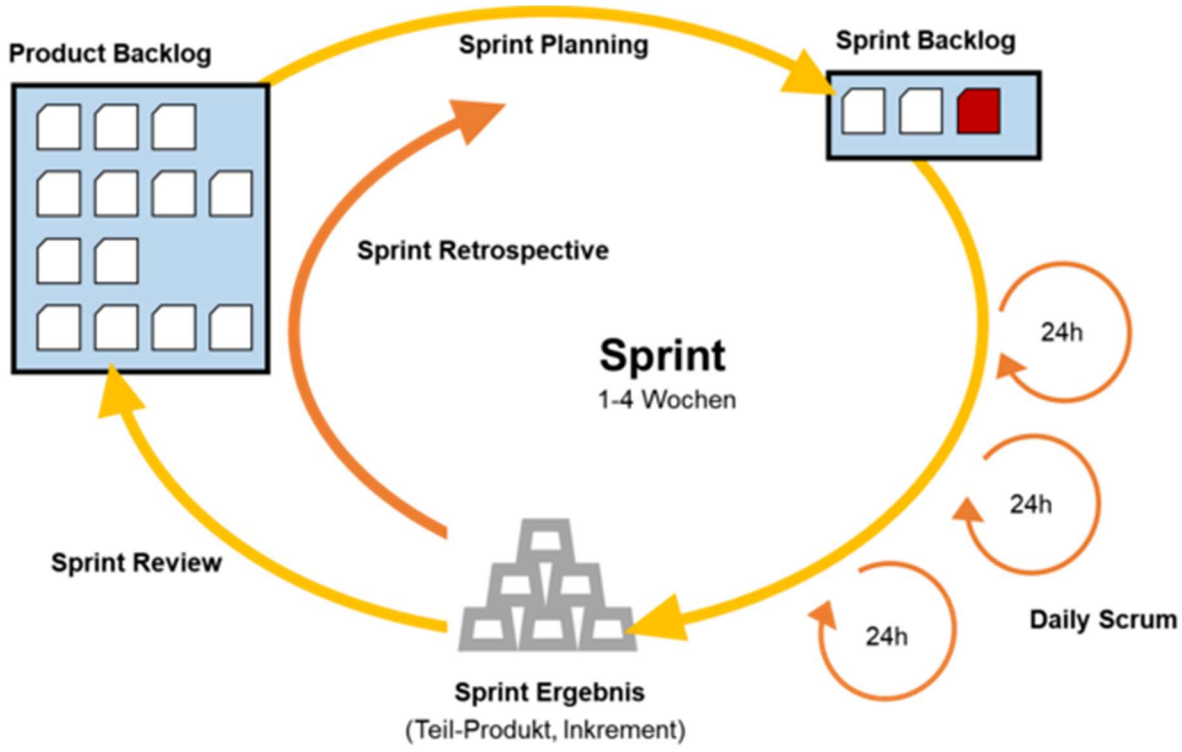
- Während des täglichen, 15-minütigen Treffens, dem sogenannten Daily Scrum, berichtet jeder der Reihe nach: Was er seit dem letzten Daily Scrum gemacht hat, was er bis zum nächsten Daily Scrum tun wird und was ihn bei seiner Arbeit behindert. Dabei haben alle das Sprint-Ziel im Blick und prüfen, ob es noch erreicht werden kann. Der Scrum-Master muss die Hindernisse aufgreifen und helfen, dass sie beseitigt werden. In einem Sprint-Burndown-Chart wird sichtbar gemacht, wie das Projekt voranschreitet.

12.4.3.6 Im Sprint Review die Sprint-Ergebnisse prüfen und abnehmen

- Jeder Sprint wird durch ein Sprint-Review-Meeting abgeschlossen. Das Team stellt die Ergebnisse und die Teil-Produkte dem Produkteigner vor. Er prüft, ob sie den Kriterien entsprechen, die mit der Definition of Done festgelegt wurden. Ist das der Fall, nimmt er die Sprint-Ergebnisse ab. Der Eintrag im Sprint Backlog ist damit abgehakt. Gleichzeitig werden die Einträge im Product Backlog aktualisiert oder angepasst.
- Mit den Ergebnissen aus dem Sprint Review und mit dem überarbeiteten Product Backlog kann der nächste Sprint starten. Der Prozess beginnt wieder mit Schritt 2. Es folgen so viele Sprints, bis das Produkt entwickelt und das Projekt abgeschlossen ist.

12.4.3.7 Mit einer Sprint Retrospective die Zusammenarbeit besprechen

- In gesonderten Treffen zwischen einem Sprint Review und dem nächsten Sprint Planning können die Teammitglieder besprechen, wie der Sprint in Bezug auf die Zusammenarbeit der beteiligten Personen, Abläufe, Kommunikation und Werkzeuge verlief. Sie halten fest, was für den nächsten Sprint verbessert werden sollte. Die Erkenntnisse sollten für zukünftige Scrum-Projekte genutzt werden; so wird ein Lernprozess unterstützt.
- Alle Schritte im Scrum-Prozess sind in der folgenden Abbildung zusammengefasst.



Scrum-Prozess und Sprint im Überblick

12.5 Fehler finden und identifizieren

- Fehler im Programm sind ein leidiges Thema – weil unvermeidbar
- Falls ein Programm Fehler enthält, müssen diese gefunden werden und beseitigt werden, bevor das Programm freigegeben wird
- Welche Fehlertypen gibt es?
    1. Typografische Fehler beim Schreiben des Programms
    2. Syntaktische Fehler beim Schreiben des Programms
    3. Fehler zur Laufzeit, die auf logische Fehler im Programmaufbau zurückzuführen sind
    4. Fehler zur Laufzeit, die auf äußere Umstände (Situationen, die erst zur Laufzeit entstehen) zurückzuführen sind
- die Beseitigung der Fehlertypen 1 bis 3 ist das, was man unter Debugging versteht
- Fehlertyp 4 lässt sich durch die Ausnahmebehandlung (Exception) abfangen
- unter einer Ausnahme (Exception) wird die Unterbrechung des normalen Programmablauf aufgrund einer besonderen Situation verstanden