Einsatz von Vorgehensmodellen

1. Allgemeines

Ein Vorgehensmodell ist ein Muster zur Beschreibung eines Entwicklungsprozesses für ein Softwaresystem. Im Rahmen eines Vorgehensmodells werden Richtlinien für folgende Teilbereiche des Software Engineering empfohlen:

- Phasen der Softwareentwicklung
- Methoden
- Rollen und Verantwortlichkeiten
- Aufgaben und Aktivitäten
- Dokumente

Seit Beginn der Geschichte des Software Engineering werden eine Reihe unterschiedlicher Vorgehensmodelle eingesetzt, denen in der aktuellen Praxis verschiedene Bedeutung zukommt. Hier die fünf wichtigsten Modelle, nach Relevanz absteigend geordnet:

- *Rational Unified Process*
  Der Schwerpunkt des Rational Unified Process liegt in der Use-Case Modellierung sowie in der Analyse und dem Entwurf eines Softwaresystems.

- *Wasserfallmodell*
  Hier steht die sequentielle und stufenweise Entwicklung eines Produktes im Vordergrund, wobei jede Aktivität bzw. Phase abgeschlossen sein muss bevor die nächste beginnt.

- *V-Modell*
  Das V-Modell ist eine um Qualitätssicherungsmaßnahmen erweiterte Version des Wasserfallmodells.

- *Extreme Programming*
  Dieses Modell ist Teil der sogenannten „Agilen Softwareentwicklung“ und beschreibt eine Vorgehensweise, die eine rasche Entwicklung von funktionsfähigen Modulen mit einem Minimum an Modellierung als Ziel hat.

- *Catalysis*
  Catalysis ist ein Vorgehensmodell für die Erstellung komponentenbasierter Software.

Eine Reihe weiterer Modelle, auf die in diesem Dokument nicht näher eingegangen wird sind: Spiralmodell, Springbrunnenmodell, Fusion, Perspective, Code And Fix,…

Motivation für den Einsatz von Vorgehensmodellen

Die Forschung und die Softwareindustrie betreiben seit Jahrzehnten beträchtliche Anstrengungen, die Produktivität und die Qualität der Softwareentwicklung zu verbessern.

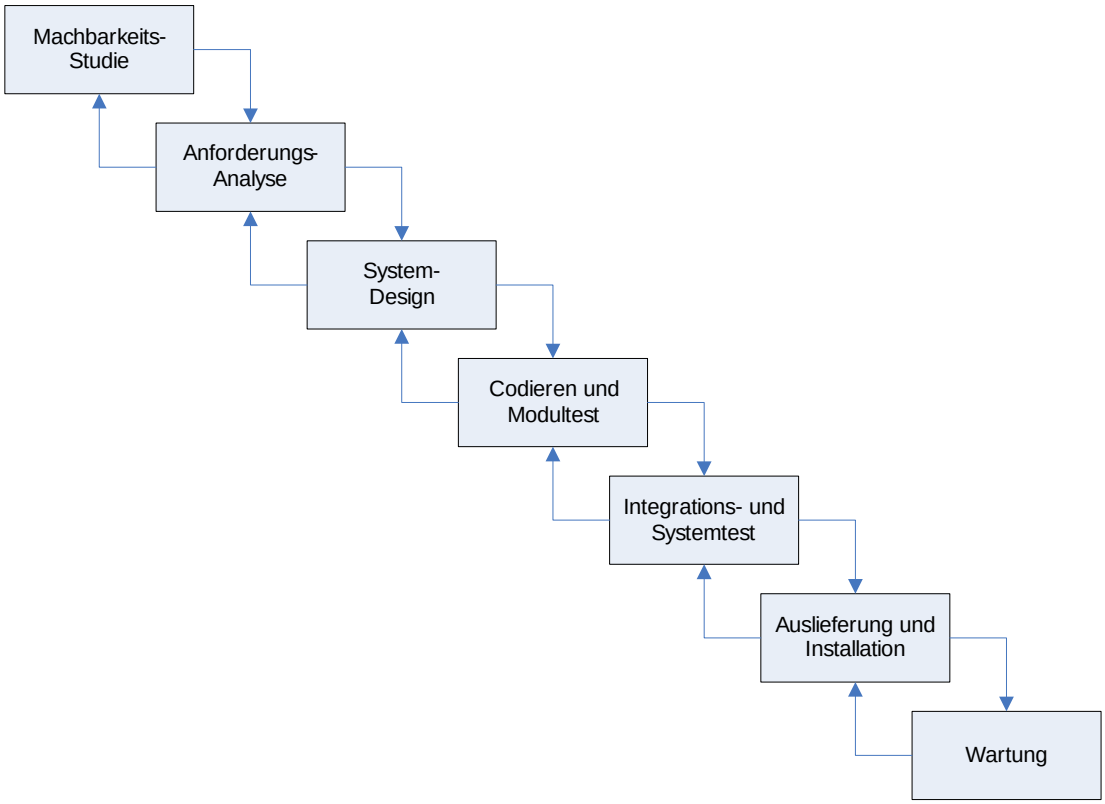
Die Entwicklung von Vorgehensmodellen spielt in diesem Prozess aus folgenden Gründen eine wichtige Rolle:

- Das Verständnis für komplexe Softwareprozesse wird erhöht.
- Eine effektivere Kommunikation über den Prozess zwischen den Anwendern, Entwicklern und den Managern wird ermöglicht.
- Softwareprojekte können besser abgeschätzt werden.
- Entwickler haben eine bessere Orientierung bei Großprojekten.
- Teilziele und Meilensteine können gesetzt und gesteuert werden.
- Projekte werden vollständig dokumentiert und werden somit analysierbar und vergleichbar.
- Prozesse können wieder verwendet werden.

2. Wasserfallmodell

Beim Wasserfallmodell werden die einzelnen Stufen des Softwareentwicklungsprozesses strikt einzeln durchlaufen. Rückkopplungen auf andere Phasen des Prozesses darf es nur auf die unmittelbar davor liegende Phase geben.

In der Fachliteratur wird man nur selten auf zwei identische Modellbeschreibungen des Wasserfallmodells treffen. Sowohl in der Einteilung der Phasen als auch in der Rückkopplungsstrategie finden sich viele Unterschiede. Eine für die Praxis geeignete Darstellung ist folgende:



Folgende Charakteristika lassen sich für das Wasserfallmodell aufzählen:

- Jede der oben skizzierten Aktivitäten wird in der vollen Breite vollständig durchgeführt.
- Am Ende jeder Phase steht ein Dokument.
- Jede Aktivität muss zuerst beendet sein, bevor die nächste beginnt.
- Rückschritte und Revisionen gibt es immer nur höchstens zur unmittelbar vorhergehenden Phase des Entwicklungsprozesses.
- Die Beteiligung der Benutzer ist lediglich in der Anforderungs-Analyse-Phase vorgesehen.

Die Phasen des Wasserfallmodells sind:

- *Machbarkeitsstudie*
  In dieser Phase wird das vorliegende Problem analysiert und informell beschrieben, wodurch ein Lastenheft (= grobes Pflichtenheft) entsteht. Erste Lösungsansätze, Kalkulationen und Kostenschätzungen führen schlussendlich zu einem Angebot.

- *Anforderungsanalyse*
  Hier werden Systemeigenschaften wie z. B. Funktionsumfang, Leistung und Schnittstellen festgelegt, wobei das „Was“ im Vordergrund steht. Das Ergebnis dieser Phase ist das Feinpflichtenheft.

- *System Design*
  In dieser Phase steht das „Wie“ im Vordergrund. Die Software-Architektur wird entwickelt und das System in Module bzw. Teilsysteme zerlegt mit diversen Design- und Entwurfsdokumenten als Output.

- *Codieren und Modultest*
  Dies ist die eigentliche Implementierungs- und Testphase, in der jedes Modul realisiert und validiert wird. Das Ergebnis dieser Phase sind Implementierungsberichte und Testprotokolle.

- *Integrations- und Systemtest*
  Die einzelnen Module werden schrittweise zu einem Gesamtsystem zusammengebaut und Gesamtsystemtests werden durchgeführt. Das Endergebnis dieser Phase ist das fertige System, die technische Dokumentation sowie die Benutzerhandbücher.

- *Auslieferung und Installation*
  Hiermit ist die Inbetriebnahme der Software beim Kunden gemeint. Dies beinhaltet auch die Schulung der Benutzer sowie Akzeptanztests.

- *Wartung*
  Das Durchführen von Fehlerbehebung und Anpassungen bzw. Erweiterungen nach der Auslieferung fällt in die Phase der Wartung.

Das Wasserfallmodell ist sehr einfach und verständlich und dadurch mit wenig Managementaufwand verbunden. Ferner hat es in der Vergangenheit als eines der ersten Prozessmodelle seiner Art wesentlich zu einem disziplinierten, sichtbaren und kontrollierbaren Prozessablauf beigetragen.

Die Probleme und Nachteile des Wasserfallmodells beruhen hauptsächlich auf dem strikt sequentiellen Ablauf der Entwicklungsschritte, der sich in der Praxis als nicht immer sinnvoll erweist. Dadurch könnten Risikofaktoren eines Projektes unterschätzt oder außer Acht gelassen werden. Ferner besteht durch den „Zwang“, am Ende jeder Phase ein Dokument abzuliefern, die Gefahr, dass die Dokumentation wichtiger als das eigentliche System genommen wird. Zu guter Letzt sei noch gesagt, dass die mangelnde Partizipation der User des zu entwickelnden Systems leicht zu Streitfällen und aufwändigen Redesigns führen kann.

3. V-Modell

Das V-Modell ist vom Wasserfallmodell abgeleitet mit der Erweiterung, dass frühe Phasen mit späteren Phasen über Testfälle verbunden sind. Wichtige Bestandteile des V-Modells sind die Verifikation sowie die Validation der Teilprodukte.

Verifikation bedeutet die Überprüfung der Übereinstimmung zwischen einem Softwareprodukt und seiner Spezifikation, während unter Validation die Eignung eines Produktes bezogen auf seinen Einsatzzweck verstanden wird.

Verifikation: „Wird ein korrektes Produkt entwickelt?“
Validation: „Wird das richtige Produkt entwickelt?“

Generell werden im V-Modell für folgende Bereiche des Softwareentwicklungsprozesses Vorgehensweisen festgelegt:

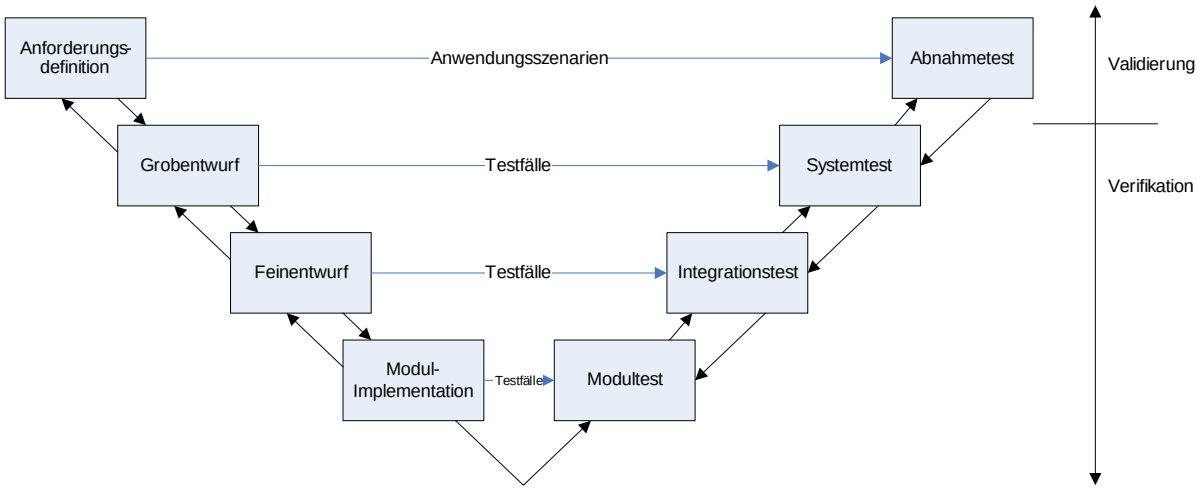
- *Systemerstellung*
  Hier werden Richtlinien für die Entwicklung und Dokumentation, sowie die Testfälle festgelegt

- *Qualitätssicherung*
  In diesem Bereich geht es um die Festlegung von Qualitätsanforderung sowie Methoden zu deren Sicherstellung

- *Konfigurationsmanagement*
  Im Konfigurationsmanagement geht es um die Verwaltung von Produkten, Rechten und Änderungen

- *Projektmanagement*
  Hier werden Vorgehensweisen zur Planung, Kontrolle, Steuerung und Koordination festgelegt.

Folgende Grafik zeigt das V-Modell aus dem Bereich der Systemerstellung:



Von der Anforderungsdefinition werden Anwendungsszenarien direkt an den Abnahmetest geleitet. Vom Grobentwurf werden Testfälle direkt an den Systemtest gegeben, ebenso wie vom Feinentwurf zum Integrationstest und von der Modulimplementation zum Modultest. Kommunikationsschritte können sowohl von der linken oberen Seite des "V" als auch von der rechten oberen Seite gestartet werden.

Das V-Modell deckt nicht nur den reinen Softwareentwicklungsprozess ab, sondern auch verwandte vor- und nachgereihte Prozesse wie z. B. Projektmanagement, Qualitätssicherung und Konfigurationsmanagement. Die einzelnen Bereiche des V-Modells beinhalten detaillierte Vorgaben, Ergebnismuster sowie Rollendefinitionen und –zuordnungen. Dadurch wird eine hoch standardisierte Abwicklung von Projekten ermöglicht. Das V-Modell ist im Gegensatz zum Wasserfallmodell auch für Großprojekte geeignet.

Der hohe Detailgrad bei den Standardisierungen des V-Modells wirft natürlich die Frage auf, ob es auch für kleinere Projekte geeignet ist, und ob es nicht zu einem Bürokratie-Overhead kommt. Zugleich besteht ein erhöhter Managementaufwand für V-Modell Projekte nicht zuletzt durch einen erhöhten Schulungsaufwand für die Projektmitarbeiter. Weiters ist die Verwendung von CASE-Tools beinahe obligatorisch, um die Vielzahl an Ergebnistypen und Prozessschritte zu modellieren.

Als weiteres Problem des V-Modells gilt die Tatsache, dass die gängigsten Versionen des Modells bestimmte Entwicklungsmethoden und –techniken fix vorgeben (Daten- und Funktionstrennung), wodurch andere Analyse- und Designmethoden unter Umständen außer Acht gelassen werden.

4. Rational Unified Process

4.1. Allgemeines

Der Unified Process ist ein Modell, das aus dem Rational Objectory Process abgeleitet wurde. Der Schwerpunkt bei diesem Basismodell sind die Use-Case Modellierung, die Analyse und der Entwurf. Unified Process ergänzt Objectory um die Themen Implementierung, Test, Anforderungs- und Konfigurationsmanagement.

Durch den Unified Process wird ein generischer Prozess-Rahmen definiert, wodurch sich der Einsatz von Unified Process auf folgende Bereiche erstreckt:

- Unterschiedliche Anwendungsbereiche
- Verschiedene Organisationen
- Unterschiedliche Kompetenzebenen
- Verschiedene Projektgrößen

4.2. Kernmerkmale des Unified Process

Die Kernmerkmale des Unified Process sind folgende:

- *Use-Case getrieben*
  Die funktionalen Anforderungen aus Sicht der Systemnutzer werden mithilfe von Use-Cases ermittelt. Auf Basis dieser Use-Cases werden Entwurfs- und Implementierungsmodelle entwickelt, und in der Endphase wird das fertige System anhand der Use-Cases getestet.

- *Architekturzentriert*
  Neben den Use-Cases kommt der Beschreibung der Systemarchitektur große Bedeutung zu. Die Architektur wird auf Basis der Use-Cases und Beschränkungen aus der Implementierungsumgebung gebildet.

- *Iterativ und inkrementell*
  Im Unified Process Modell wird ein großes Softwaresystem in kleinen Schritten oder Mini-Projekten realisiert. Jedes dieser Mini-Projekte ist eine Iteration und liefert als Ergebnis ein Inkrement. Jede einzelne Iteration umfasst eine Gruppe von Use-Cases für die die Entwicklungsschritte
  - Anforderungsermittlung
  - Analyse
  - Entwurf
  - Implementierung
  - Test

  durchlaufen werden. Jede folgende Iteration baut auf den Ergebnissen der vorhergehenden Iteration auf (vgl. Wasserfallmodell).

- *Komponentenorientiert*
  Das Vorgehensmodell des Unified Process ist komponentenbasiert. Dadurch wird ein Softwaresystem aus Softwarekomponenten zusammengesetzt, die über klar definierte Schnittstellen verbunden sind. Nach Unified Process sind Komponenten physikalische und austauschbare Teile eines Systems, die Schnittstellen bereitstellen und realisieren.

4.3. Phasen des Unified Process

Die Entwicklung eines Softwaresystems kann entweder in einem Stück oder in mehreren Zyklen ablaufen, wobei am Ende jedes Zyklus die Auslieferung eines Produktrelease an den Kunden steht.

Jeder Zyklus besteht aus 4 Phasen:

- Beginn
- Ausarbeitung
- Konstruktion
- Übergang

In jeder dieser Phasen werden die im vorherigen Kapitel erwähnten Iterationen der folgenden Arbeitsabläufe durchgeführt:

- Anforderungsermittlung
- Analyse
- Entwurf
- Implementierung
- Test

In der ersten Phase (**Beginn**) wird zunächst die Machbarkeit des Systems beurteilt. Es kommt zu einer ersten Formulierung von Use-Cases, wodurch ein erster Eindruck der Funktionalität des Systems entsteht. Möglicherweise kommt es sogar zu einer ersten Prototyperstellung, um die Kernideen des Systems zu veranschaulichen. Die Hauptaktivitäten der Beginnphase sind die folgenden:

- *Definition der Grenzen des Systems*
  Durch die Festlegung von Zielen und expliziten Nicht-Zielen soll die Abschätzung des Gesamtumfanges erleichtert werden.

- *Beschreibung einer möglichen Ergebnisarchitektur*
  Hier erfolgt eine erste Beschreibung der technischen Architektur des zu entwickelnden Software-Systems.

- *Erste Identifikation von Risiken*
  Risiken, die die Realisierbarkeit des Systems gefährden, sollen möglichst in der Beginnphase identifiziert werden. Hier können auch mögliche Reaktionen zum Umgang mit eventuell eintretenden Risiken definiert werden.

- *Präsentation des vorgeschlagenen Systems (am Papier oder Prototyp)*
  Um zu Beginn sicherzugehen, ob der Kunde und der Softwarehersteller das gleiche Verständnis über die Aufgabenstellung und die mögliche Lösung haben, ist eine Vorstellung des Lösungsansatzes in der Beginnphase unerlässlich.

Die Phase der **Ausarbeitung** hat die Vorbereitung und Planung der nächsten Phase (Konstruktion) zum Ziel. Hier werden folgende Aktivitäten beinhaltet:

- *Entwicklung einer Architektur*
  Diese sollte stabil sein und die architekturrelevanten Funktionalitäten des Systems beinhalten. Es werden Modelle entwickelt und eventuell erste Prototypen entwickelt.

- *Festlegung von Qualitätsattributen*
  Hier werden konkrete Qualitätsmerkmale definiert, die vom Software-System erreicht werden sollen.

- *Dokumentation von mindestens 80 % der funktionalen Anforderungen*

- *Planung der Konstruktionsphase*
  Das bedeutet Planung von Inkrementen und Zuordnung von Ressourcen zu den Inkrementen.

Ziel der **Konstruktionsphase** ist die Realisierung der vorhin geplanten Inkremente. Folgende Aktivitäten werden im Rahmen dieser Phase ausgeführt:

- *Ausführliche Beschreibung bereits identifizierter Use-Cases*

- *Abschluss der Systemanalyse*

- *Wartung der bereits existierenden Architektur*

- *Risikokontrolle und Einleitung von gezielten Gegenmaßnahmen*

Mit der Phase des **Übergangs** wird das System aus der Entwicklungsumgebung in die Benutzerumgebung transferiert. Hierunter fallen folgende Aktivitäten:

- *Vorbereitung des Einsatzortes der Software*

- *Anweisungen an den Kunden zur Aktualisierung der Einsatzumgebung*

- *Bereitstellung von Benutzerhandbüchern und anderen Dokumentationen*

- *Anpassung der Software an die Benutzerumgebung*

- *Behebung von aufgedeckten Fehlern*

4.4. Aktivitäten und Ergebnisse des Unified Process

Der Unified Process unterscheidet zwischen fünf Kernarbeitsabläufen:

- Anforderungsermittlung
- Analyse
- Entwurf
- Implementierung
- Test

In der **Anforderungsermittlung** werden die Anforderungen an ein Software-System erarbeitet und dokumentiert. Dieser Kernarbeitsablauf geschieht in vier Schritten:

- *Auflistung potenzieller Anforderungen*
  In diesem Schritt werden alle möglichen Anforderungen der verschiedensten Systemnutzer ermittelt und dokumentiert, wobei eine Liste mit so genannten Features angelegt wird, von denen jedes benannt und kurz beschrieben wird.

- *Aufbau eines Verständnisses für den Systemkontext*
  Ziel dieser Aktivität ist es, die Systemumgebung zu verstehen. Hier bietet sich einerseits die Modellierung der Geschäftsprozesse an, beziehungsweise bei eingebetteten Systemen eine Modellierung der Domäne.

- *Ermittlung der funktionalen Anforderungen*
  Das Ermitteln der funktionalen Anforderungen umfasst eine Reihe von Teilaktivitäten. Einerseits müssen zunächst die Use-Cases und die Akteure identifiziert werden. Die ermittelten Use-Cases werden dann priorisiert, um eine Architektursicht auf wichtige Use-Cases zu liefern. Im nächsten Schritt werden die Use-Cases etwas detaillierter beschrieben, insbesondere jene Information, die zwischen den Akteuren und dem System ausgetauscht wird. Zustands-, Aktivitäts- oder Sequenzdiagramme können der Output dieses Arbeitsschrittes sein. Abschließend kann es sich als opportun erweisen, erste Prototypen der Benutzerschnittstellen zu entwerfen.

- *Ermittlung der nicht-funktionalen Anforderungen*
  Hier sind Anforderungen an spezielle Systemeigenschaften gemeint, die sich aus Umgebungs- oder Implementierungsbeschränkungen ergeben, z. B. Performance. Teilweise lassen sich auch nicht-funktionale Anforderungen den einzelnen Use-Cases zuordnen.

Im Kernarbeitsablauf **Analyse** werden die erhobenen Anforderungen analysiert um daraus ein Modell zu entwickeln. Ziel ist es, die dokumentierten Anforderungen zu verfeinern und zu strukturieren, um die Anforderungen besser zu verstehen und eine wartbare Systemstruktur zu erhalten. Während dieser Phase werden vier Hauptaktivitäten unterschieden:

- *Architekturanalyse*
  Hier werden Analyse-Teilsysteme identifiziert, die für die Architektur des Software-Systems relevant sind. Ergebnis ist eine Architekturbeschreibung.

- *Use-Case Analyse*
  Ziel dieser Aktivität ist es, Analyseklassen zu identifizieren und Interaktionen mittels Kollaborationsdiagrammen zu beschreiben. Zusätzlich können die Interaktionen textuell beschrieben werden.

- *Analyse der Klassen*
  Während dieser Aktivität werden Verantwortlichkeiten, Eigenschaften, Beziehungen und spezielle Anforderungen für einzelne Klassen mit Hilfe von Klassendiagrammen beschrieben.

- *Analyse des Packages*
  Hier werden Teilsysteme analysiert, die das Analysesystem in kleinere verwaltbare Stücke aufteilt.

Die dritte Entwicklungsaktivität ist der **Entwurf**. Die Definition einer Systemstruktur, die alle funktionalen und nicht-funktionalen Anforderungen einschließt ist das Ziel dieser Phase. Während des Entwurfs sollte gewährleistet sein, dass das System in verwaltbare Teile aufgeteilt wird, die von verschiedenen Entwicklungsteams parallel bearbeitet werden können. Aus dem Entwurf sollte direkt die Implementierung ableitbar sein. Hier lassen sich vier Aktivitäten unterscheiden:

- *Architekturentwurf*
  Hier wird nun eine Architektur für das Software-System festgelegt und in diversen Diagrammen festgehalten.

- *Use-Case Entwurf*
  In diesem Schritt sollen Teilsysteme oder Entwurfsklassen identifiziert werden, um den Ereignisfluss innerhalb des Systems zu modellieren.

- *Klassenentwurf*
  Der Entwurf einer Klasse umfasst die Festlegung verschiedener Aspekte. Es werden Operationen, Attribute, Beziehungen, Zustände und Abhängigkeiten für jede Klasse festgelegt. Ferner wird beschrieben, wie die Schnittstellen korrekt realisiert werden.

- *Teilsystementwurf*
  Hier soll gewährleistet werden, dass ein Teilsystem so unabhängig wie möglich von anderen Teilsystemen ist. Alle Teilsysteme müssen die richtigen Schnittstellen zur Verfügung stellen, um korrekt realisiert zu werden.

Während der Phase der **Implementierung** wird das System basierend auf dem Entwurf in die Realität umgesetzt. Sämtliche Entwurfsklassen werden kodiert und getestet. Hierbei lassen sich fünf Aktivitäten unterscheiden:

- *Architekturimplementierung*
  Dieser Schritt definiert eine Struktur für das Implementierungsmodell durch die Identifikation jener Komponenten, die für die Architektur relevant sind.

- *Systemintegration*
  Ziel dieser Aktivität ist die Entwicklung eines Integrationsbauplans, womit beschrieben wird, welche Bauteile für eine Iteration benötigt werden.

- *Implementierung der Teilsysteme*
  Hier wird sicher gestellt, dass ein Teilsystem jene Rolle im Endsystem einnimmt, die im Integrationsbauplan vorgesehen wurde.

- *Klassenimplementierung*
  Die Implementierung der Klassen hat zum Ziel, den Entwurf einer Klasse in Form einer Dateikomponente zu realisieren. Dazu gehört die Kodierung und Implementierung der Eigenschaften und Methoden einer Klasse sowie die Überprüfung der Schnittstellen.

- *Komponententest*
  Während dieser Aktivität wird die Struktur und das Verhalten einer Klasse getestet. Im Rahmen eines Strukturtests wird die interne Implementierung der Komponente überprüft, während beim Verhaltenstest das Verhalten einer Komponente getestet wird.

Die letzte Entwicklungsaktivität ist der **Test**, dessen Ziele die Überprüfung des Ergebnisses der Implementierung sowie der Kommunikation zwischen den Bauteilen des Software-Systems sind. Es lassen sich hier sechs Teilaktivitäten unterscheiden:

- *Testplanung*
  Hier soll der Testaufwand für eine Iteration geplant werden. Dazu gehört die Abschätzung der Anforderungen an die Ressourcen, die Verteilung des Testaufwandes und die Entwicklung einer Teststrategie für die Iteration.

- *Testentwurf*
  In diesem Arbeitschritt werden Testfälle festgelegt und Testprozeduren definiert, wodurch festgelegt wird, wie Tests ausgeführt werden.

- *Testimplementierung*
  Nach Möglichkeit sollen Testprozeduren automatisiert werden, was in dieser Phase geschieht.

- *Durchführung Integrationstests*
  Die zuvor definierten Testfälle werden durchgeführt und die Testergebnisse dokumentiert. Die Testergebnisse werden mit den erwarteten Ergebnissen verglichen und eventuelle Abweichungen werden dokumentiert und untersucht.

- *Durchführung Systemtests*
  Die definierten Systemtests werden in diesem Schritt durchgeführt und die Ergebnisse erneut dokumentiert. Mit dieser Aktivität kann begonnen werden wenn der Integrationstest die definierten Qualitätsziele bestätigt.

- *Testbewertung*
  Hier soll die Bewertung des Testaufwands innerhalb einer Iteration erfolgen. Die Testergebnisse werden mit den Testzielen des Testplans verglichen.

4.5. Anpassung und Einführung

Das Modell des Unified Process ist ein Rahmenwerk und muss daher an verschiedene Umgebungsbedingungen angepasst werden. Diese sind z.B.:

- Systemgröße
- Domäne
- Komplexität

Das Vorgehensmodell gibt nur einen Überblick über die Arbeitsprozesse. Für die konkrete Anwendung ist weiteres Wissen notwendig wie z.B. Einsatz von Werkzeugen, CASE-Tools oder konkrete Handlungsanweisungen.

Wie bei der Einführung anderer Vorgehensmodelle erfordert die Einführung von Unified Process konkrete Einführungsstrategien. In diese Strategien müssen vor allem die wichtigsten Umgebungsfaktoren einbezogen werden. Die Einführung könnte in folgenden drei Schritten ablaufen:

1. *Einführung fundamentaler Prinzipien und Techniken*

   Hier werden eventuell existierende Vorgehensmodelle auf den Einsatz von Unified Process abgestimmt, was nicht bedeuten muss, dass alte und bewährte Methoden über Board geworfen werden. Vielmehr muss überprüft werden ob die bisherigen Vorgehensweisen gegen fundamentale Prinzipien des Unified Process verstoßen.

2. *Einsatz des Modells*

   In dieser Stufe erfolgt die vollständige Abänderung der Vorgehensweise auf Unified Process. Die Entwicklungsphasen, Techniken, Abläufe und Werkzeuge werden übernommen und eingesetzt. Nach dieser Phase steht eine funktionsfähige Umgebung zur Verfügung.

3. *Optimierung und Anpassung des Modells*

   In der letzten Phase werden die Anpassung der Prozesse an die lokale Umgebung sowie diverse Optimierungstätigkeiten vorgenommen.

Ergänzend zu diesem Text steht Ihnen auf der InfraSoft Website ein Glossar zur Verfügung, in dem Sie die meisten der hier verwendeten Begriffe finden. Über das aktuelle Angebot an weiteren, kostenlosen Fachbeiträgen zur Softwareentwicklung informieren Sie sich bitte unter www.infrasoft.at/service.

Mag. Andreas Fandl
Wien, im Mai 2005

Der Autor ist Mitarbeiter der InfraSoft, einem Unternehmen, das auf komplexe Softwareentwicklungen spezialisiert ist. Die Experten der InfraSoft haben langjährige Erfahrungen in der Entwicklung und verfügen über fundierte Kenntnisse in Design, Analyse, Realisierung, Test und Projektmanagement.
Für **individuelle Beratungen** zur Entwicklung von Softwarelösungen und die Bereitstellung von **Realisierungsteams** wenden Sie sich bitte an info@infrasoft.at.