

## Ausgabe 2003 / 2

Erscheinungsart: ca. 4 x jährlich in elektronischer Form

# Vorgehensmodelle

### weitere in dieser Ausgabe ...

- ⇒ Vorgehensmodelle - Eine Leine für den 'Werwolf' Software?
- ⇒ Das Spiralmodell von Pomberger und Pree
- ⇒ Vorgehensmodelle im Überblick
- ⇒ Interessante Links zum Thema

### Kurzdefinition / Glossar ...

- ⇒ Vorgehensmodelle sollen den Arbeitsprozess vereinheitlichen, um eine verlässliche Arbeitsgrundlage zu schaffen. [Informatik-Handbuch]
- ⇒ Vorgehensmodelle sind eine einheitliche und verbindliche Vorgabe für Aktivitäten und deren Ergebnisse. [V-Modell97]

### Vorgehensmodelle: 'Kochrezepte' für die Software-Entwicklung ...

Man kann natürlich auch ohne Kochrezepte kochen. Mit entsprechendem Wissen und praktischer Erfahrung wird dies auch meist mehr oder weniger gut gelingen. Das Ergebnis ist jedoch oft nicht vorhersagbar (manchmal gelingt es, manchmal ist es etwas 'versalzen' und teilweise ist es auch ungenießbar). Mit einem Kochrezept, das je nach Bedarf ev. auch bewusst etwas abgewandelt werden kann, wird ein gleichmäßigeres Ergebnis erzielt werden können (auch noch nicht so erfahrene Köche werden damit zumeist Erfolg haben).

Welchen Nutzen bringt es nun, wenn man in der Software-Entwicklung Vorgehensmodelle definiert und anwendet?

- ⇒ **Eindämmung der Kosten über den Lebenszyklus eines Software-Produkts** (reduzierter Ressourceneinsatz - frühes Erkennen von Fehlentwicklungen - geringere Schulungskosten - transparentere Kalkulation - geringeres Kostenrisiko - ...)
- ⇒ **Qualitätsverbesserung** (Standardisierung - einfachere Prüfung - einfachere Wartbarkeit - ...)
- ⇒ **Verbesserung der Kommunikation** (einheitliche Begriffe - standardisierte Zwischenergebnisse - detailliertere Beschreibungen - weniger Missverständnisse - ...)
- ⇒ **Zusatznutzen** (oft schon gute Grundlage für angestrebte Zertifizierungen oder Auditierungen wie z.B. ISO9000, CMM, ...).

### Nutzen und Tipps für Software-Käufer / Kunden ...

Auch Software-Käufer bzw. Interessenten können das Wissen über Vorgehensmodelle zum eigenen Nutzen verwenden:

- ⇒ **Indiz für Qualität:** wenn ein Lieferant ein adäquates Vorgehensmodell verwendet, ist dies zumindest schon ein Indiz für Professionalität. (Einen Vergleich gängiger Vorgehensmodelle finden Sie weiter hinten.)
- ⇒ Lassen Sie sich das Vorgehensmodell Ihres Lieferanten vor einer Entscheidung genau erklären und ziehen Sie wenn notwendig einen IT-Spezialisten hinzu. Sie werden einerseits Ihre Lieferanten und auch deren Preisbildung besser verstehen und andererseits sehr schnell die **Spreu vom Lieferanten-Weizen trennen** können.
- ⇒ **Gute Basis für Zusammenarbeit:** der Lieferant und auch Sie als Käufer wissen, wie und in welchen Schritten die Leistungen erbracht werden und wann welches Ergebnis zu erwarten ist.



**Theorie und Praxis  
bei Vorgehensmodellen**

"Eine Entscheidung ist besser als keine Entscheidung", "Ein Businessplan ist besser als kein Businessplan", ...

Man könnte noch unzählige Beispiele finden, in denen die grundsätzliche Notwendigkeit nach einer Planung und systematischen Vorgehensweise hervorgehoben und unterstrichen wird.

Im IT-Bereich ist es nicht anders.

Leider jedoch wird in IT-Unternehmen oder auch IT-Abteilungen oft auf klare Vorgehensmodelle verzichtet.

Es wird dem Einzelnen überlassen, sich mehr oder weniger gut zu organisieren.

Manchmal funktioniert dies auch erstaunlich gut. Zumeist jedoch verursacht eine sehr individuell gesteuerte Vorgehensweise unmittelbar erheblich erhöhte Prozesskosten durch Reibungsverluste, Wissensverluste bei Weggang/Neuzugang von Mitarbeitern, ... ganz zu schweigen von den erhöhten Wartungs-, Support- und Betriebskosten von qualitativ mangelhaften Systemen.

Viele Verantwortliche meinen, sich kurzfristig einen Kostenvorteil verschaffen zu können, wenn sie auf klare Vorgehensweisen und Regelungen verzichten.

Dass dies nicht so ist, zeigt die Praxis und wird auch in vielen Studien belegt, in denen die im IT-Bereich dramatisch hohe Anzahl fehlgeschlagener Projekte aufgezeigt wird.

### Dipl.-Ing. Johannes Bergmann

allgemein gerichtlich beideter und  
zertifizierter Sachverständiger für Informatik

Der Quality-Newsletter ist ein periodisches Informationsmedium von Software Quality Lab und dessen Partnern mit Schwerpunkt Software-Qualitätsmanagement.

Inhalt: fachliche Beiträge und Schwerpunktthemen, Vorstellung neuer Produkte und Leistungen, neue wissenschaftliche Erkenntnisse, Branchen-News, ...

Aktuelle Fach- und Forschungsbeiträge sind willkommen.

Einsendungen an [info@software-quality-lab.at](mailto:info@software-quality-lab.at).

Weitere Infos zu diesem und anderen Themen finden Sie auf <http://www.software-quality-lab.at>.

## Vorgehensmodelle: Eine Leine für den 'Werwolf' Software?

von Univ.-Prof. Dr. Gerhard Chroust

Abteilung für Systemtechnik und Automation, Institut für Systemwissenschaften, Johannes Kepler Universität Linz

Software hat einige Eigenschaften, die die Sicherstellung ihrer Qualität schwieriger machen als bei vielen anderen Industrieprodukten: Software ist primär ein Gedankenprodukt, ist unsichtbar, sehr komplex, leicht änderbar und soll sich jeder Umgebung leicht anpassen (Fred Brooks, 1968).

**Kleinste Veränderungen im Code** (z.B. ein fehlerhafter Strichpunkt) **können bereits schwerwiegendste Änderungen des Verhaltens hervorrufen** (wie zum Beispiel den Absturz der Ariane-Rakete).

Fred Brooks sprach deshalb auch vom 'Werwolf Software', den man mit keiner noch so magischen Silberkugel erlegen kann.

Auf Grund der Komplexität der Software-Produkte ist es chancenlos, die Korrektheit von Software am End-Produkt durch Tests u.ä. festzustellen. Man kann den Werwolf Software nur an die Leine legen, indem man den Entwicklungsprozess definiert, den die Entwickler anwenden müssen: wie man ein Kind an der Hand über eine gefährliche Straße führt.

**Vorgehensmodelle sind eine Abstraktion von bestehenden, erfolgreichen Entwicklungsprozessen** (ähnlich einer Klasse in der objekt-orientierten Programmierung), die den Ablauf von zukünftigen Software-Entwicklungsprozessen festlegen, wobei aber der Prozess für jedes individuelle Projekt an die äußeren Projektparameter (Aufwand, Termin, Können der Entwicklungsmannschaft, Qualitätsanforderungen, Risiko) anzupassen ist.

Die bindende Festlegung eines definierten Vorgehensmodells bietet einige **wesentliche Vorteile**:

- Ein Vorgehensmodell abstrahiert vom individuellen Projekt und legt somit die generelle Entwicklungsmethodik (die 'Software-Kultur') fest.
- Es erzwingt die korrekte Abfolge der notwendigen Entwicklungsschritte und sichert die Einheitlichkeit von Methoden und Werkzeugen im Projekt.
- Es erlaubt eine verbesserte vorausschauende Planung.
- Es macht die Entwicklung vom individuellen Entwicklungsingenieur unabhängig.
- Die Lehr- und Lernbarkeit des in einem Unternehmen vorgegebenen Entwicklungsprozesses wird erleichtert.
- Der Prozess kann (unabhängig vom konkreten Produkt) theoretisch betrachtet, analysiert und durch Erfahrung zu verbessert werden.

- Es dient auch zur Bewertung und Zertifizierung eines Software-Hauses im Sinne von ISO 9000, CMM oder Bootstrap.

In ihrer Grundkonzeption bestehen Vorgehensmodelle aus der Beschreibung der Aktivitäten und der von den Aktivitäten erzeugten und verwendeten Daten, sowie in der Festlegung der Ausführungsreihenfolge dieser Aktivitäten.

Dadurch wird auch im wesentlichen die 'Strategie' der Software-Entwicklung festgelegt: in welcher Reihenfolge die Aktivitäten (eventuell auch mehrfach) durchlaufen werden.

Die aktuell propagierten **Vorgehensmodelle unterscheiden sich**

- durch den **Detaillierungsgrad**: vom einfachen Boehm'schen Wasserfall-Phasenmodell bis zu detailliert ausgearbeiteten Modellen wie V-Modell oder Hermes
- durch die Festlegung der **Sequentialisierungsreihenfolge** der Entwicklungsschritte: vom linearen Durchlauf wie z.B. beim Wasserfall-Modell bis zu komplexen, iterativen Ansätzen wie bei Spiralmodell und Rational Unified Process (RUP).
- durch die Berücksichtigung der **begleitenden Prozesse**: keine bei Wasserfall-Modell und RUP, Qualitätsmanagement und Projektmanagement bei IBM's ADPS und zusätzlich Konfigurationsmanagement beim V-Modell.

Heute eingesetzte Vorgehensmodelle müssen vor ihrem effektivem Einsatz an die Entwicklungsorganisation und ihre etablierten Konventionen und Usancen (die 'Software-Kultur') angepasst werden.

Sie müssen auch in eine entsprechende Software-Entwicklungsumgebung (Software Engineering Environment) eingebettet werden, die sowohl methoden-kompatible Entwurfswerkzeuge (z.B. UML-Diagramme) als auch entsprechende Verwaltungsmittel für Zwischen- und Endergebnisse bietet.

Vorgehensmodelle sind ein **wesentlicher Beitrag zur qualitativen Verbesserung des Software-Entwicklungs-Prozesses**, indem sie Systematik, Vollständigkeit und korrekte Abfolge der einzelnen Aktivitäten weitgehend sicherstellen.



# Vorgehensmodelle im Überblick

von Dipl.-Ing. Johannes Bergmann

Es gibt eine Vielzahl von unterschiedlichen Vorgehensmodellen im Software-Engineering-Umfeld. Es gibt Vorgehensmodelle, die sich ausschließlich auf die Phasen der Software-Entwicklung konzentrieren. Umfassendere Vorgehensmodelle betrachten auch andere wesentliche Bereiche bzw. Prozesse wie z.B. Projektmanagement und Qualitätsmanagement.

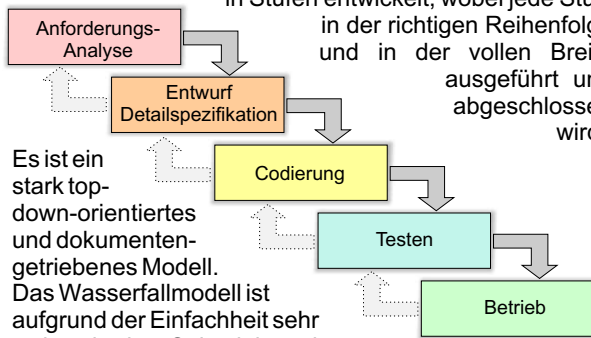
In der Literatur und in Publikationen finden sich immer wieder einige Vorgehensmodelle, die sich in der Praxis etabliert haben und/oder eine bestimmte Klasse von Vorgehensweisen zusammenfassen (abstrahieren).

Nachfolgend sind einige wesentliche Vorgehensmodelle im Überblick angeführt und kurz charakterisiert:

- **Code & Fix:** Das Urmodell besteht aus 2 Schritten:
  1. Schreibe ein Programm
  2. Finde und behebe die Fehler im Programm
 Nachteile: keine Entwurfsphase - das Programm wurde schnell unleserlich und die Fehlerbehebung teuer - die Benutzer akzeptieren ein Programm nicht, das sie nicht selbst (mit-)spezifiziert haben - ...  
 Leider wird diese 'Methode' des Software-Engineerings (wenn auch meistens in leicht 'entschärfter' Version) heute noch immer in vielen Projekten angewendet!

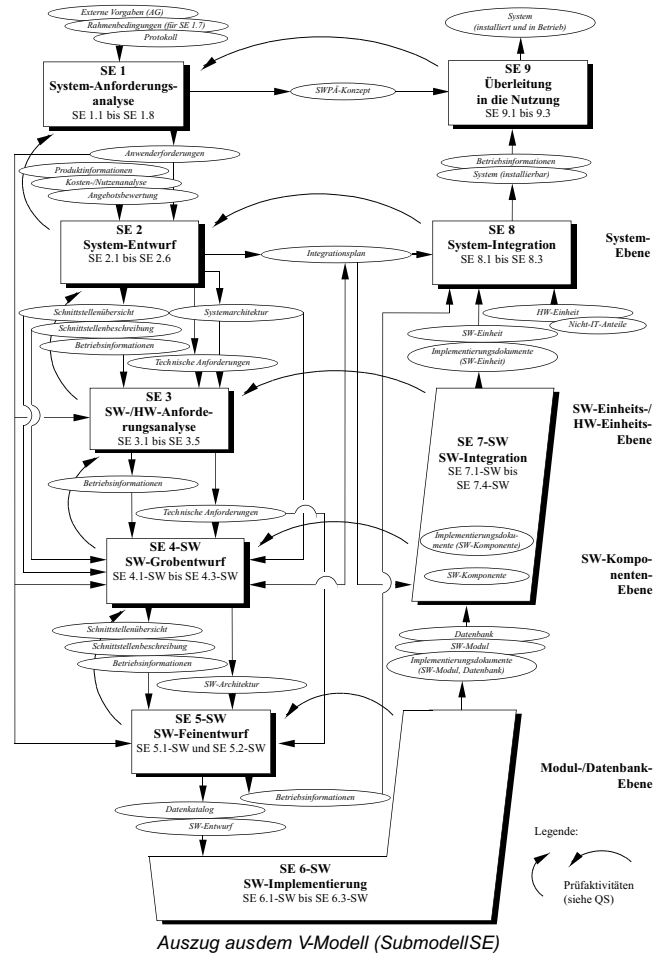
## Klassische Modelle

- **Wasserfall-Modell:** Die Software wird dabei sukzessiv in Stufen entwickelt, wobei jede Stufe in der richtigen Reihenfolge und in der vollen Breite ausgeführt und abgeschlossen wird.



Es ist ein stark top-down-orientiertes und dokumentengetriebenes Modell. Das Wasserfallmodell ist aufgrund der Einfachheit sehr weit verbreitet. Sehr viele andere Modelle wurden davon abgewandelt. Wesentliche Nachteile sind jedoch die sehr starre (wenig flexible) sequentielle Durchführung sowie fehlende Bereiche wie z.B. Berücksichtigung von Projektmanagement.

- **V-Modell:** Das V-Modell (derzeit noch in der Version '97) ist der Entwicklungsstandard für IT-Systeme der deutschen Bundesbehörden (in einer leicht abgewandelten Fassung auch für die österreichische Verwaltung). Das V-Modell ist sehr umfangreich und berücksichtigt neben dem eigentlichen Software-Entwicklungsprozess (SE) auch weitere Bereiche wie Projektmanagement (PM), Qualitätsmanagement (QM) und Konfigurationsmanagement (KM) die miteinander verknüpft sind. Grundsätzlich ist es sehr breit und umfassend ausgeführt und kann (soll) durch Anpassung ('Tayloring') an die jeweilige Projektgröße und -situation angepasst werden.



Das V-Modell ist ein sehr komplexes Modell. Es stellt jedoch auch für Software-Entwickler, die es nicht in seiner Gesamtheit anwenden (wollen), einen guten Fundus dar für Dokumentvorlagen und Checklisten in den verschiedensten Bereichen des Software-Engineerings.

Da das V-Modell schon etwas in die Jahre gekommen ist, wurde Ende 2002 ein Weiterentwicklungsprojekt initiiert das unter Federführung der TU-München das 'V-Modell 200x' entwickeln soll. Dies soll bis Ende 2004 abgeschlossen sein. Als wesentliche Erweiterungen sollen auch Themen wie neue Methoden des Software-Engineerings sowie Faktoren wie Ergonomie, Usability, ... berücksichtigt werden, eine Trennung in Produktmodell und Prozessmodell erfolgen und die Konformität zum Standard 'CMM(1)' angestrebt werden.

- **Prototypen-Modell:** Der Auftraggeber ist oft nicht in der Lage, seine Anforderungen vollständig zu spezifizieren. Manchmal ist es auch notwendig, verschiedene Lösungsmöglichkeiten experimentell zu erproben oder die Realisierbarkeit von Anforderungen vor der Fertigstellung des Systems zu untersuchen. Um diesen Anforderungen gerecht zu werden, wurden Prototypen-Modelle entwickelt.

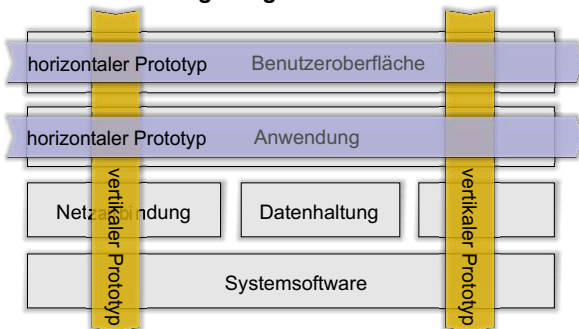
Fortsetzung auf nächster Seite >>>

Fortsetzung - Vorgehensmodelle Überblick >>>

Eine Übersicht über einige gebräuchliche Prototypen-Arten ist nachfolgend angeführt:

- **Demonstrationsprototyp:** Soll einen ersten Eindruck vermitteln und dient z.B. zur Unterstützung in der Akquisition. Es ist oft ein schnell erstellter (rapid prototyping) Wegwerfprototyp.
- **Prototyp im engeren Sinn:** parallel zur Anwendungsmodellierung erstellt, um Benutzerschnittstelle oder Teil-Funktionen zu veranschaulichen.
- **Labormuster (experimenteller Prototyp):** unterstützt die Beantwortung konstruktionsbezogener Fragen (z.B. Architektur des Systems, Laufzeitverhalten, ...).
- **Pilotsystem (evolutionärer Prototyp):** Ist selbst schon der Kern des Produkts. Die Unterscheidung zum Produkt verschwindet. Ab einem gewissen Reifegrad geht der Prototyp automatisch in das Produkt über.

Weitere Unterscheidungsmöglichkeiten:

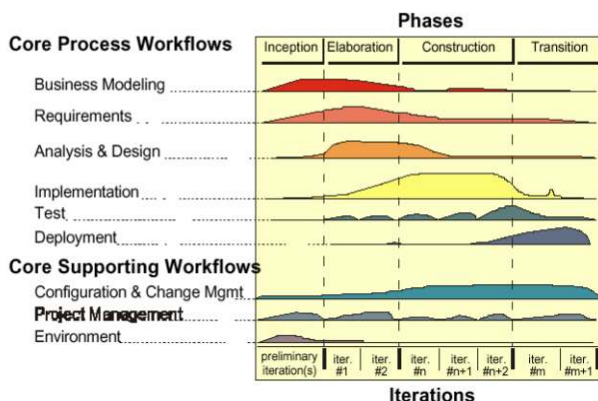


- **Spiralmodell:** Das Spiralmodell, von Boehm Ende der 80er entwickelt, sowie abgeleitete neuere Varianten und Erweiterungen (siehe auch 'das Spiralmodell von Pomberger und Pree') stellen Metamodelle dar. Für jede Stufe sind im Wesentlichen 4 Schritte zu durchlaufen:
  1. Zieldefinition, Abwägen von Alternativen, zu beachtende Randbedingungen festlegen.
  2. Evaluierung der Möglichkeiten, Erkennen und Vermeiden von Risiken.
  3. Entwicklung und Verifikation des Produkts der nächsten Stufe.
  4. Planung der nächsten Phase(n).

Neuere Modelle

→ **RUP - Rational Unified Process:**

RUP wurde von der Firma Rational (seit kurzem im IBM-Konzern) entwickelt und wird sehr gut durch die umfangreiche Software-Familie von Rational unterstützt.



RUP ist ein umfangreiches generisches Framework, verwendet sehr intensiv die Modellierungssprache UML und ist damit für objektorientierte Entwicklungen gut geeignet.

RUP basiert auf 2 grundsätzlichen Dimensionen:

- Phasen: Management-Sicht zur Beurteilung des Fortschritts.
- Workflows: arbeitsteilige Sicht auf die einzelnen Tätigkeiten.

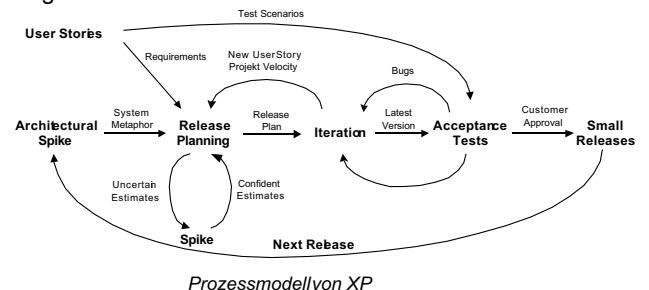
In die Entwicklung von RUP sind Erfahrungswerte aus vielen Software-Projekten eingeflossen: Iterative Softwareentwicklung, Anforderungsmanagement als Grundlage für den Funktionsumfang, komponentenbasierte Architektur, visuelle Software-Entwicklung, kontrollierte Software-Qualität, ...

Ähnlich wie bei anderen großen Vorgehensmodellen sollte auch der RUP an die jeweilige Projektsituation entsprechend angepasst werden.

→ **Extreme Programming (XP):**

eXtreme Programming (XP) zählt zu den leichtgewichtigen Methoden bei der Softwareentwicklung. Es stellt hohe Anforderungen an die Anwender, um seine Wirkungsweise voll zu entfalten. Wesentliche Eigenschaften des Modells sind:

- Kurze Iterationszyklen mit testbaren Resultaten.
- Keep it simple: das einfachste Design, das ausreicht, wird implementiert.
- Ständiges Testen der Entwickler: Zu jeder Klasse wird eine eigene Test-Klasse entwickelt. Empfohlen ist sogar, zuerst die Testfälle zu entwickeln und dann erst die Anwendung!
- Sehr kurze Auslieferungs- und Release-Zyklen
- Refactoring: Ständiges Überarbeiten und Vereinfachen der Architektur, ohne die Funktionalität zu verändern.
- PairProgramming: Immer 2 Programmierer arbeiten gemeinsam am Programm.
- Collective ownership: jeder kann den Code jederzeit ändern
- 40-Stunden-Woche: niemals in 2 Wochen hintereinander Überstunden machen!
- Coding Standards: gemeinsame Code-Richtlinien
- On-Site-Customer: Ein oder mehrere Mitarbeiter des Auftraggebers sind in das Entwicklerteam eingebunden.
- Untergeordnete Rolle der Dokumentation bis hin zum gänzlichen Verzicht.



XP bietet interessante Ansätze, stellt jedoch sehr hohe Anforderungen hinsichtlich (Selbst-)Disziplin der Programmierer, des Managements und auch des Kunden. Geeignete Anwendungsbereiche sind speziell Situationen, in denen sich die Anforderungen an das zu implementierende System in kurzen Abständen ändern. Problematisch an diesem Prozessmodell ist, dass einige der zugrundeliegenden Prinzipien bei zunehmender Projektkomplexität immer weniger geeignet sind.

Fortsetzung auf nächster Seite >>>

Fortsetzung - Vorgehensmodelle Überblick >>>

## ➤ Aktueller Trend: Agile Softwareentwicklung

Neben dem relativ bekannten eXtreme Programming Ansatz existieren noch weitere agile Methoden, die in den letzten Jahren entwickelt und verbreitet wurden:

- AD - Agile Database Techniques
- AM - Agile Modeling
- ASD - Adaptive Software Development
- Crystal
- FDD - Feature Driven Development
- DSDM - Dynamic Systems Development Method
- Lean Software Development
- Scrum
- TDD - Test-Driven Design
- XBreed

Nachfolgend werden einige dieser Modelle kurz skizziert:

Alistair Cockburn vertritt den Crystal-Ansatz, eine auf Menschen und Kommunikation fokussierte Sichtweise der Softwareentwicklung. Je nach Risikostufe (Kritikalität) und der Anzahl der involvierten Personen wird eine bestimmte passende Ausprägung des Prozessmodells angewendet.

Jim Highsmith ist der Erfinder von Adaptive Software Development (ASD) und arbeitet inzwischen eng mit Alistair Cockburn zusammen.

Feature Driven Development wurde von Jeff DeLuca entwickelt. In den fünf Phasen des FDD sind die einzelnen Features eines Softwareprojektes der zentrale Dreh- und Angelpunkt. Features sind keine technischen Eigenschaften sondern kleine, für den Kunden geschäftswertige Funktionen.

Scrum sieht ein empirisches Prozessmodell als Weiterentwicklung iterativ-inkrementeller Vorgehensweise vor. Einige Prinzipien der Komplexitätstheorie werden hier auf die Softwareentwicklung übertragen. In Scrum werden regelmäßig "Sprints" durchgeführt, in denen die eigentliche Entwicklung stattfindet. Der Fokus liegt hier auf dem Management von kleinen Teams, weniger auf technischen Details.

Weiters gibt es noch den Ansatz 'Pragmatic Programmer', der eine eigene Stilrichtung der leichtgewichtigen Entwicklungsmethoden darstellt und eng mit dem eXtreme Programming verwandt ist.

Allen 'agilen' Ansätzen gemeinsam ist, dass sie sich in der Agile-Alliance auf die wesentlichen Grundsätze agiler Softwareentwicklung geeinigt haben.

Im 'The Agile Software Development Manifesto', das Anfang 2001 von 17 bekannten Persönlichkeiten aus dem Bereich des Software-Engineerings verfasst wurde, sollen bessere Wege zur Software-Entwicklung aufgezeigt werden.

Es wurden 4 (sicherlich zu hinterfragende) Kernpunkte definiert:

- 1.) Individuals and interactions OVER process and tools
- 2.) Working software OVER comprehensive documentation
- 3.) Customer collaboration OVER contract negotiation
- 4.) Responding to change OVER following a plan

Problematisch an den Ansätzen der agilen Software-Entwicklung ist vor allem, dass sie von einem Idealbild ausgehen, das in der Praxis oft nicht anzutreffen ist:

Kunden, die sich teilweise auch mit mehreren Mitarbeitern sehr intensiv und permanent mit dem Projekt beschäftigen und die die Leistungen der Entwickler möglichst nach tatsächlich erbrachtem Aufwand honorieren und nicht mehrere Anbieter finanziell gegeneinander ausspielen.

Lieferanten, die selbstlos und ehrlich die für den Kunden beste Softwareentwicklung anstreben und die den eigenen finanziellen Nutzen nach hinten stellen.

Aufgrund der teilweise zu idealistischen Ansätze ist davon auszugehen, dass einige der aktuellen Vorgehensmodell-Entwicklungen als Mode-Erscheinungen sehr schnell wieder in Vergessenheit geraten werden.

## Resümee:

- Das ideale, alle Situationen abdeckende Vorgehensmodell gibt es nicht.
- Aus der Vielzahl an Vorgehensmodellen sollte ein oder ev. auch einige (für unterschiedliche Situationen) passende Modelle ausgewählt werden.
- Die Vorgehensmodelle sollten dann an die jeweilige Organisations- oder Projektsituation angepasst werden, um eine entsprechende Wirksamkeit zu erreichen und Ineffizienz und zu vermeiden.
- Die ausgewählten Vorgehensmodelle müssen mit der Organisation, den Mitarbeitern und ev. auch den Kunden weiterentwickelt werden und ständig an die sich verändernden Rahmenbedingungen angepasst werden.

## Zusätzliche Infos ...

### Links zum Thema:

- ⇒ **V-Modell und Ö.-Bundesvorgehensmodell**  
(abgeleitet v. deutschen V-Modell)  
<http://www.ansstand.deh> , <http://www.bv-modell.at/>
- ⇒ **(Rational) Unified Process, UML**  
<http://www.rational.com> <http://www.omg.org>
- ⇒ **eXtreme-Programming**  
<http://www.extremeprogramming.org/>
- ⇒ **Agile Programmierung**  
<http://www.agilealliance.org>

### Weitere nützliche Links:

- ⇒ [http://www.cetus-links.org/oo\\_ooa\\_ood\\_methods.html](http://www.cetus-links.org/oo_ooa_ood_methods.html)  
Sehr umfangreiche Sammlung an Links rund um objekt-orientierte Methoden und Modelle (auch wenn manche Links ins Leere zeigen)
- ⇒ <http://www.rational.com/media/whitepapers/ad2000-05-05.pdf>  
OMG-Whitepaper zum Thema Unified-Process-Model
- ⇒ <http://www.stickyminds.com>  
Webseite mit interessanten Inhalten zu fast allen Themen der Software-Entwicklung

### Impressum:

Herausgeber und für den Inhalt verantwortlich:

Die Beiträge wurden sorgfältig ausgewählt und bearbeitet.  
Für Druckfehler und Irrtümer wird nicht gehaftet.

### Software Quality Lab

Fliederstrasse 8  
A-4222 Langenstein

[www.software-quality-lab.at](http://www.software-quality-lab.at)

[info@software-quality-lab.at](mailto:info@software-quality-lab.at)

Tel.: +43-(0)664-16 20 220, Fax: +43-(0)7237-4941