# Software Engineering - Der Software Life Cycle

## Inhaltsverzeichnis



## 1 Ziele

Durch das Software-Engineering werden Methoden aus dem Bereich der Ingenieurwissenschaften für die Vorgehensweise der Softwareentwicklung adaptiert. Der Softwareentwicklungsprozess soll dadurch plan- und kalkulierbar werden.

Definition des Softwareengineering nach W. Hesse:

Softwareengineering ist das Fachgebiet der Informatik, das sich mit der Bereitstellung und systematischen Verwendung von Methoden und Werkzeugen für die Herstellung und Anwendung von Software beschäftigt.

## 2 Vorgehensmodelle für die Softwareentwicklung

Vorgehensmodelle sind Arbeitspläne, die den Projektverlauf der Softwareentwicklung strukturierend gestalten, um im Ergebnis eine hochwertige, gut dokumentierte Software zu erhalten. Im folgenden werden vier alternative Vorgehensweisen beschrieben: das *Wasserfallmodell*, das *Rapid Prototyping,* das *Spiralmodell* und *Business Reenginering*.

### 2.1 Wasserfallmodell (Phasenmodell)

Das Wasserfallmodell unterteilt Softwareprojekte in sechs Phasen: *Systemdefinition*, *Analyse*, *Design*, *Implementierung* sowie *Test* und *Wartung* des Softwaresystems. Am Ende jeder Phase stehen Dokumente, zum Beispiel Lastenhefte und Projektpläne, die an die folgenden Projektphasen weitergegeben werden und als Grundlage für die weitere Entwicklung dienen.

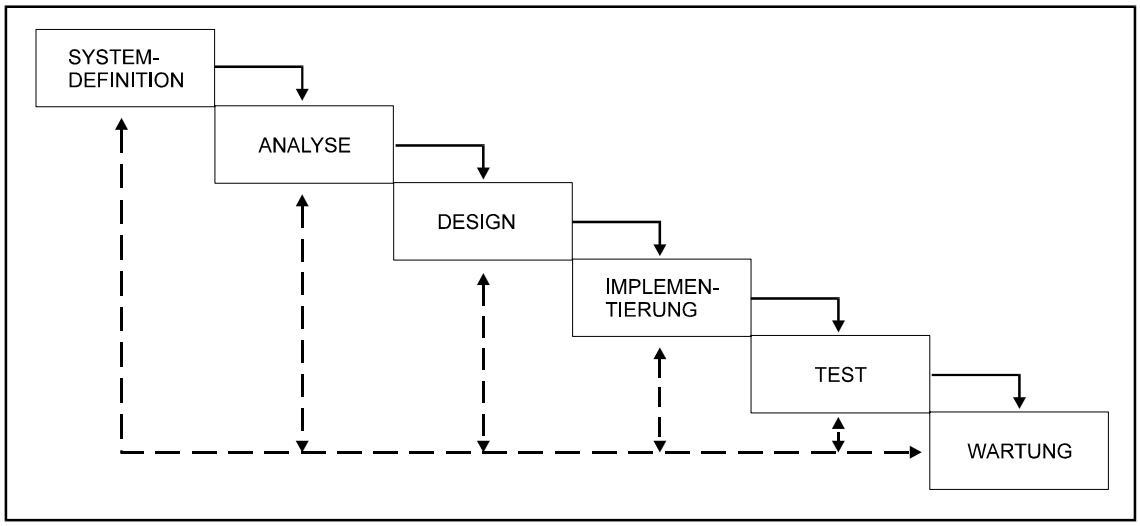


Bild: Das Wasserfallmodell

An der *Systemdefinition* sowie der *Anforderungsanalyse* sind die Fachabteilungen und die Softwareentwickler gemeinsam beteiligt. In den beiden ersten Phasen geht es darum, die Integration des Softwaresystems in die Unternehmung zu gewährleisten. Die *Analysephase* soll zusätzlich klären, wie die Aufgaben aussehen, die das System auszuführen hat, und welche Daten dazu notwendig sind. Im Vordergrund steht die Frage, was das System leisten soll.

In der *Designphase* wird die Architektur der Programme festgelegt. Diese Phase beschäftigt sich mit der Frage, wie das Softwaresystem realisiert wird, welche Module wie aufgerufen werden und welche Schnittstellen Verwendung finden.

Als vierte Stufe im Wasserfall erfolgt die *Programmierung.* Die Implementierung der Module wird erst in Angriff genommen, nachdem die fachlichen Anforderungen und

der Aufbau des Programms feststehen. Im Anschluß an die Programmierung wird das fertige Programm getestet (*Testphase*) und die Installation vorgenommen.

Die Vorteile des Wasserfallmodells liegen im geringen (geplanten) Aufwand bei Test und Wartung. Dies soll dadurch erreicht werden, dass erst nach detaillierter Planung mit der Implementierung begonnen wird. Theoretisch sieht das Wasserfallmodell keine Rückkehr zu vorangegangenen Phasen vor. In der Praxis finden jedoch Rücksprünge statt, um entstandene Fehler zu beseitigen oder Planungsungenauigkeiten zu korrigieren.

Am Phasenmodell der Softwareentwicklung wurde häufig und detailliert Kritik geübt. Die Kritikpunkt sind im folgenden zusammengestellt:

- Die älteste Kritik am klassischen Phasenmodell richtet sich gegen die strikt sequentielle Abfolge der Einzelphasen. Softwareentwicklung ist kein wasserfallartiger Prozess, sondern jede Phase kann zu Rücksprüngen auf frühere Phasen führen. Jede Produktdefinition, jede Softwarespezifikation, jede Implementation enthält Fehler, die in späteren Phasen entdeckt werden und zu rückwirkenden Änderungen führen. Insofern geht die Annahme, dass jede Phase vollständig abgeschlossen ist, bevor die nächste Phase beginnt, an der Entwicklungspraxis vorbei.

- Unpräzise oder fehlerhafte Produktdefinitionen, die an den Vorstellungen des Auftraggebers vorbeigehen, können im Phasenmodell nicht frühzeitig erkannt werden. Es wird realitätsfremd von einer abgeschlossenen und korrekten Produktdefinition ausgegangen.

- Eine lauffähige Version des Systems liegt im Phasenmodell erst am Ende der Entwicklung vor. Viele Fehler und Missverständnisse aus frühen Phasen werden dadurch sehr spät entdeckt, wodurch deren Beseitigung sehr aufwendig und teuer werden kann.

- Auftraggeber können nur in den seltensten Fällen ihre Programmvorstellungen, Anforderungen und Wünsche präzise angeben. Detaillierte Vorstellungen reifen häufig erst mit dem Fortschreiten der Entwicklung heran. Hiervon betroffen sind sowohl Gestaltungswünsche an die Benutzungsoberfläche als auch Einzelfunktionalitäten des Systems. Das Phasenmodell sieht aber eine Beteiligung des Auftraggebers über die erste Phase hinaus nicht vor.

- Bei Nichteinhalten der geplanten Projektdauer fällt im Phasenmodell das Projektende in die Testphase und führt, wenn keine Verlängerung möglich ist, zu mangelhaft getesteten Produkten. Dadurch erhöht sich die Fehlerhaftigkeit großer Softwaresysteme, da große Teile des Systemtests de facto in die Wartungsphase verlagert werden. Zwar wird dieser Tatbestand durch Managementfehler und nicht durch das Phasenmodell an sich hervorgerufen, doch werden die Auswirkungen durch das Phasenmodell verstärkt.

Die teilweise doch sehr fundamentale Kritik am Wasserfallmodell der Softwareentwicklung hat dazu geführt, alternative Vorgehensweisen zu entwickeln, die vor allem als realitätsnäher zu kennzeichnen sind.

## 2.2 Rapid Prototyping

Das Prototypenmodell eignet sich besonders für den Entwurf von dialoggesteuerten Softwaresystemen. Prototypen bestehen lediglich aus Masken für Menüs oder für die Bildschirm- und Druckerausgabe, die mit dem Anwender abgestimmt werden, jedoch keine Funktionen enthalten. Funktionale Prototypen verfügen außerdem über einen Teil des Funktionsumfangs, den das Softwaresystem später besitzen soll. Die Fragen nach Qualität und Leistung stehen in den Anfangsphasen zunächst im Hintergrund. Softwareentwicklung nach dem Prototypenmodell beginnt mit einer *Anforderungsanalyse*. Dann werden in der *Definitionsphase* die Funktionen und Daten des Systems beschrieben. Hierzu werden Datenflussdiagramme und das konzeptuelle Datenschema verwendet. Es folgen *Entwurf* und *Realisierung* des Prototypen, der nur wesentliche Merkmale des Softwaresystems berücksichtigt. Dafür kommen besonders Programmiersprachen der vierten Generation (4GL) in Frage. Diese zeichnen sich durch graphische Unterstützung bei der Programmierung aus. Mit 4GL Programmierumgebungen müssen Bildschirmmasken nicht mehr programmiert, sondern können gezeichnet werden. Der Prototyp wird dann mehrfach mit dem Anwender abgestimmt und auf der Basis dieser Gespräche weiterentwickelt. Ziel des Prototypenmodells ist es, Lücken der Benutzeroberfläche oder der Funktionalität bereits während der Entwicklungsphase auszumerzen.

Die Kritik am Prototypenmodell liegt vor allem im Aufwand für die Entwicklung und Pflege der Prototypen. Die Prototypen werden meist in sehr anwendernahen Systemen (Datenbanken, Basic-Dialekte) erstellt, deren Mächtigkeit nicht ausreicht, um große und komplexe Projekte abzuwickeln. Dies führt dazu, dass die Prototypen in vielen Projekten über die Systemdefinition hinaus keine Verwendung finden und zum Beginn der eigentlichen Programmierung (z.B. in C++) noch einmal komplett neu entwickelt werden müssen. Dieser Doppelaufwand lässt sich nur vermeiden, wenn schon bei der Prototypenentwicklung genügend mächtige Programmierumgebungen eingesetzt werden, die zwar eine sehr schnelle Masken- und Menüdefinition zulassen, sich jedoch zusätzlich noch beliebig erweitern lassen.

Vor allem neuere objektorientierte Programmierumgebungen scheinen hier geeignete Technologien bereitzustellen, um das Rapid Prototyping noch attraktiver zu machen, da hier die Prototypen tatsächlich die Basis für die weitere Entwicklung darstellen.

## 2.3 Das V-Modell

Das V-Modell stellt eine Erweiterung des Wasserfall-Modells dar, es integriert die Aspekte der Qualitätssicherung.

- Detaillierte Phasen sind versetzt unter der vorhergehenden Phase angeordnet
- Phase und zugehörige Testphase können als eine Hierarchieebene betrachtet werden
- In das Modell sind Validierung (Gültigkeitsprüfung) und Verifikation (Korrektheitsnachweis) integriert

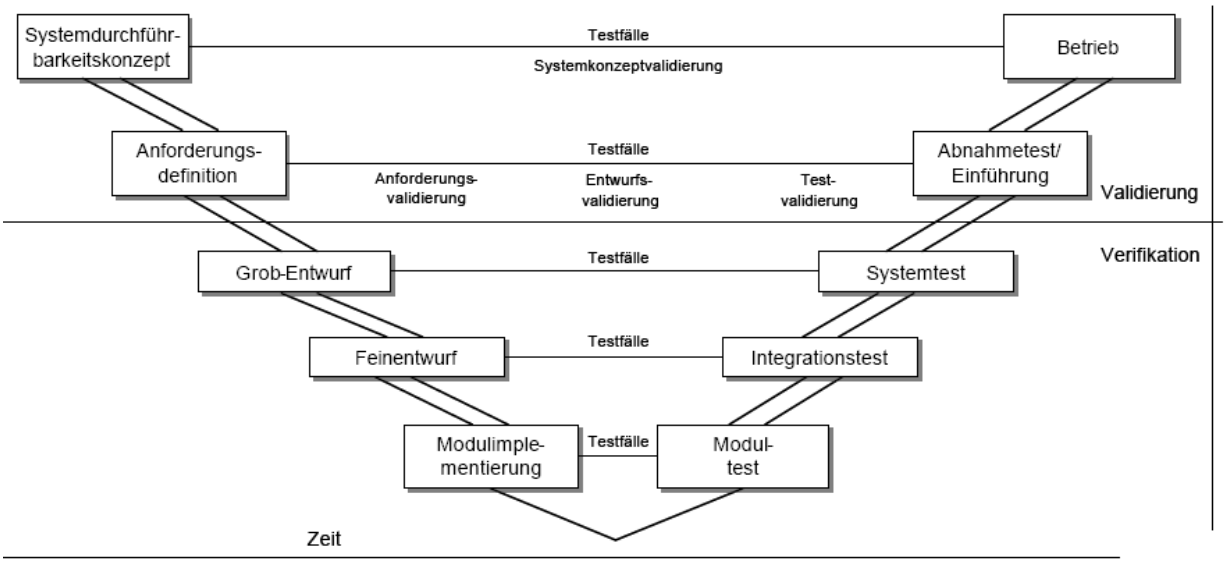


Bild: Das V-Modell

**Vorteile:**

- Gesamt-Modell (umfassendes Modell)
- Gut geeignet für große Projekte
- Qualitätssicherung steht im Vordergrund

**Nachteile:**

- Sehr generisch, Vorgehensweisen sind sehr allgemein und müssen angepasst werden
- Für kleine Projekte ungeeignet (zu viele Zwischenprodukte)
- Sehr dokumentenzentriert

## 2.4 Spiralmodell

Während das Wasserfallmodell mit der Dokumentation der Einzelphasen arbeitet, wird das Spiralmodell mittels Risikoanalyse gesteuert. Vom Nordwest-Quadranten ausgehend beschreibt es den Ablauf von Softwareprojekten als spiralförmigen Verlauf in einem Koordinatensystem.

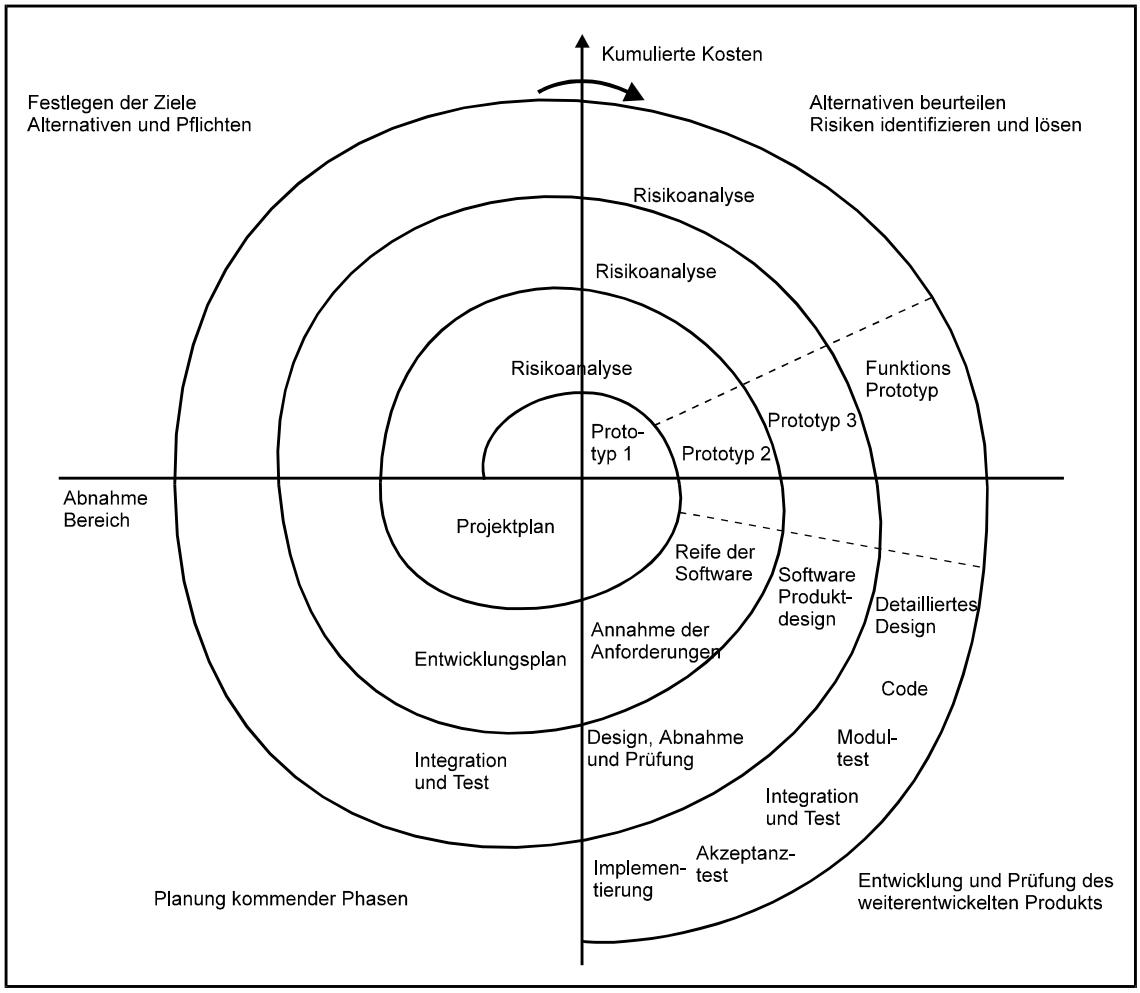

Bild: Das Spiralmodell

Die Softwareentwicklung beginnt im Spiralmodell mit der Zieldefinition und der Aufstellung von Alternativen. Im zweiten Schritt werden die Alternativen mit Hilfe von Risiko- und Wirtschaftlichkeitsanalysen beurteilt. Danach folgen ein Prototyp sowie die Weiterentwicklung und Prüfung der Software.

In jeder Drehung der Spirale sind folgende Fragen zu erläutern, die sich zum Projektende hin immer weiter konkretisieren und das fertige Anwendungssystem beschreiben.

1. Ziele, Alternativen und Rahmenbedingungen müssen festgelegt werden.
2. Die Alternativen müssen evaluiert werden, um Risiken soweit wie möglich reduzieren zu können.
3. Die realisierten Zwischenprodukte müssen überprüft und abgestimmt werden.
4. Die Projektfortsetzung muss geplant werden.

Diese Schritte werden so lange wiederholt, bis das Softwaresystem fertig gestellt ist. Das Spiralmodell bietet den Vorteil, dass Fortgang und Wirtschaftlichkeit des Softwareprojekts regelmäßig, nämlich mit jeder Spiralumdrehung, überprüft werden. Die Ausdehnung der Spirale zeigt den Grad der Ausarbeitung und die dabei entstandenen Kosten an.

*Ziele, Alternativen und Rahmenbedingungen.* Zu Beginn jedes Zyklus werden zunächst inhaltliche Vorgaben an das zu entwickelnde Teilprodukt festgelegt (z.B. Funktionalität, Qualitätskriterien, Nutzung von Bibliotheken zur Wiederverwendung) und alternative Vorgehensweisen herausgearbeitet (z.B. unterschiedliche Analyse- oder Entwurfstechniken, Einsatz von Werkzeugen, "make or buy"-Entscheidung). Besonders zu beachten sind Einschränkungen bzgl. Zeit, Personal, Kosten, Hard- und Softwareumgebungen. Die Spirale wird durch das Aufstellen der *Hypothese,* dass ein spezifischer Ablauf (im eigenen Unternehmen oder beim Kunden) durch eine Softwarelösung zu verbessern ist, initialisiert. Die Formulierung dieses Ziels und die Entwicklung konkreter Lösungsvorschläge bilden den Beginn des ersten Spiralzyklus.

*Evaluierung der Alternativen und Reduktion der Risiken.* In der zweiten Phase der aktuellen Windung werden die Alternativen hinsichtlich der festgelegten Ziele und Einschränkungen untersucht und bewertet. Es tauchen in der Regel Fragen, Unschärfen und Unwägbarkeiten auf, die jede Entscheidung für oder wider eine Alternative (bzw. die Verwertung aller Alternativen) zu einer unsicheren Entscheidung machen. Unsichere Entscheidungen sind aber Risikoquellen des Projekts. Der Grad der Unsicherheit (das Risiko) sollte daher - immer unter Berücksichtigung der zugehörigen Kosten - soweit wie möglich reduziert werden. Dazu können so unterschiedliche Techniken wie z.B. Prototyping, Simulation, Datenmodellierung, Benchmark-Tests, Marktforschung mittels Umfragen oder Literaturrecherchen eingesetzt werden. Die weitere Projektarbeit im allgemeinen und die Entscheidung für die Aktivitäten der dritten Stufe der aktuellen Windung im besonderen gründen sich nun auf das noch verbliebene Restrisiko. Dabei steht stets die Minimierung des Restrisikos im Mittelpunkt des weiteren Vorgehens.

*Realisierung und Überprüfung.* Im dritten Schritt der aktuellen Windung wird die gewählte Alternative unter Einhaltung der Ziel- und Ressourcenvorgaben realisiert und getestet. Methodisch kann unter Berücksichtigung des Restrisikos flexibel und unter Einsatz des gesamten Software Engineering-Repertoires - von evolutionärem Vorgehen über transformationelle Entwicklung bis zur verstärkten Orientierung auf Wiederverwendung - vorgegangen werden. Die Entscheidung für eine Methode (bzw. eine sinnvolle Kombination mehrerer Vorgehensweisen) erfolgt ausschließlich risikoorientiert und exklusiv für die aktuelle Windung. Sie ist in keiner Weise bindend für die nächsten Zyklen.

*Planung, Review und Planungskonsens.* Zum Abschluss der aktuellen Windung wird auf der Grundlage des aktuellen Projektstands die nächste Spiralwindung inhaltlich und organisatorisch geplant. Die Planung muss nicht auf die unmittelbar folgende Phase beschränkt bleiben, sondern kann vielmehr mehrere Windungen betreffen.

Erlaubt ist auch die Aufteilung des Projekts in weitgehend unabhängige Teilprojekte, die von verschiedenen Entwicklungsteams parallel durchgeführt und erst zu einem späteren Zeitpunkt integriert werden. Nach diesen Vorarbeiten greift der Kontrollmechanismus des Spiralmodells: In einem Review werden die Projektfortschritte des letzten Zyklus analysiert, die Ergebnisse bewertet und die Projektperspektiven diskutiert, bis unter allen beteiligten Parteien Konsens über die Situation des Projektes besteht. Sind z.B. die technischen oder wirtschaftlichen Risiken einer Projektfortsetzung (immer betrachtet vor dem Hintergrund der Anfangshypothese) zu hoch, so endet die Spirale (und das Projekt) an diesem Punkt. Erfolgt kein vorzeitiger Projektabbruch, so liegt am Ende der letzten Windung die neue oder modifizierte Software installiert vor und im letzten Review erfolgt ein abschließender Test der Anfangshypothese auf Basis des realen Ablaufs.

Das Spiralmodell stellt eine detailliert ausgearbeitete Methode des Rapid Prototyping dar, die aufgrund der Risikoanalyse eine sehr vorsichtige Planung großer Softwareprojekte zulässt und daher in der Praxis weit verbreitet ist.

## 2.5 Der Iterative Softwareentwicklungsprozess (RUP)

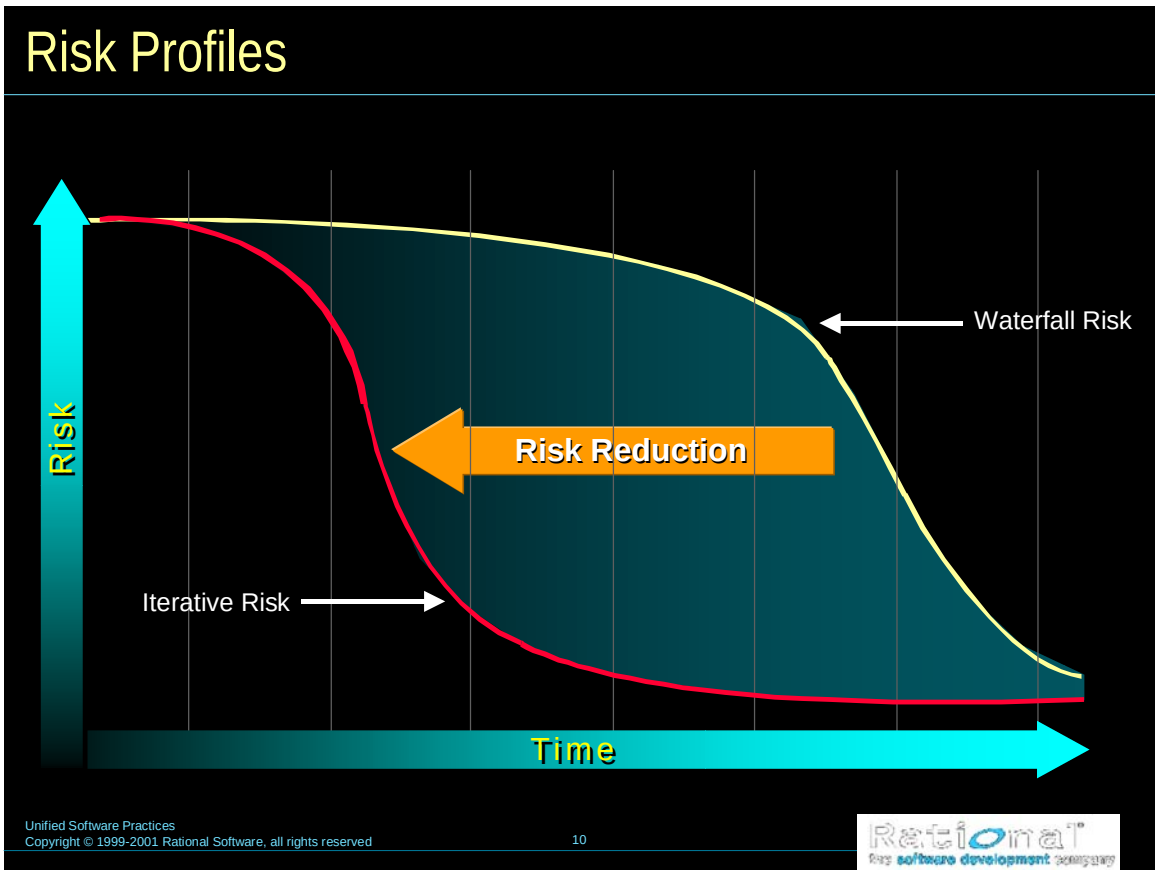


Bild: Risiken bei der Vorgehensweise nach dem Wasserfallmodell

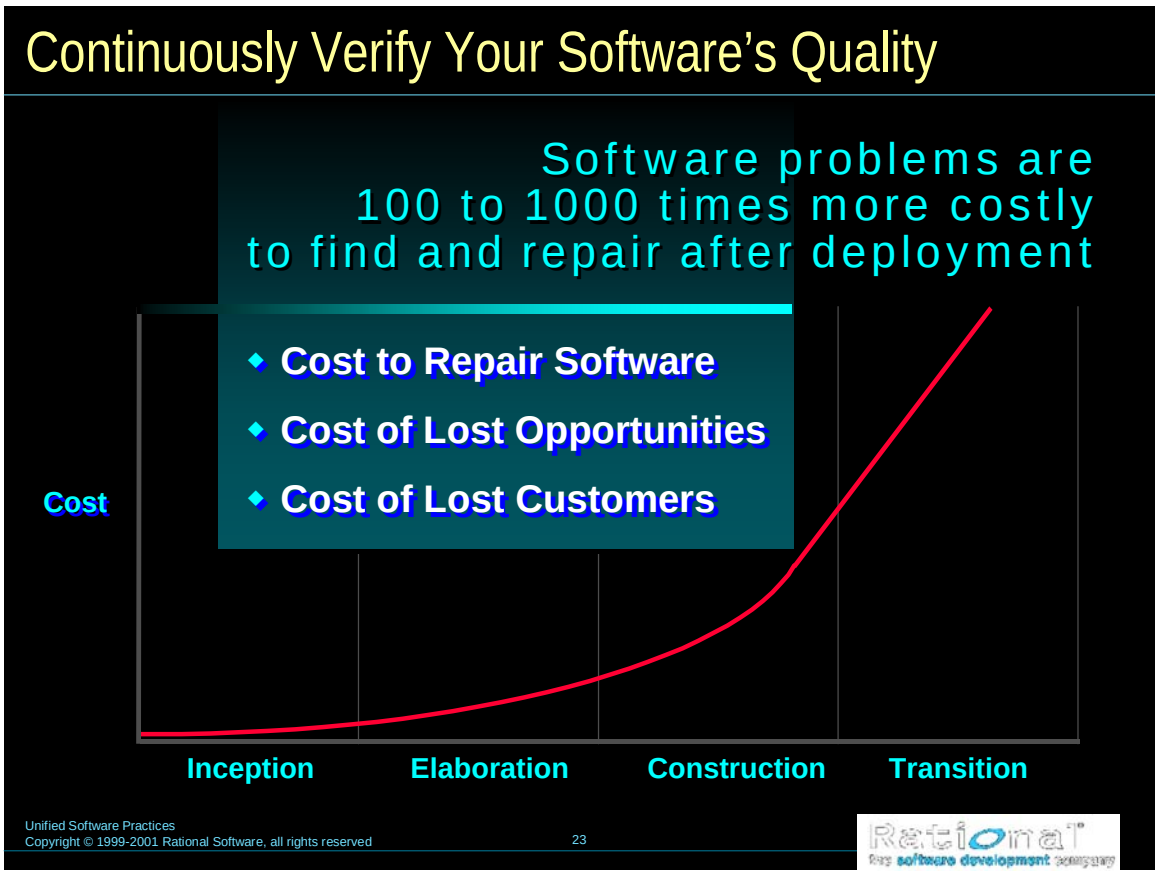


Bild: Risiken bei der Vorgehensweise nach dem Wasserfallmodell

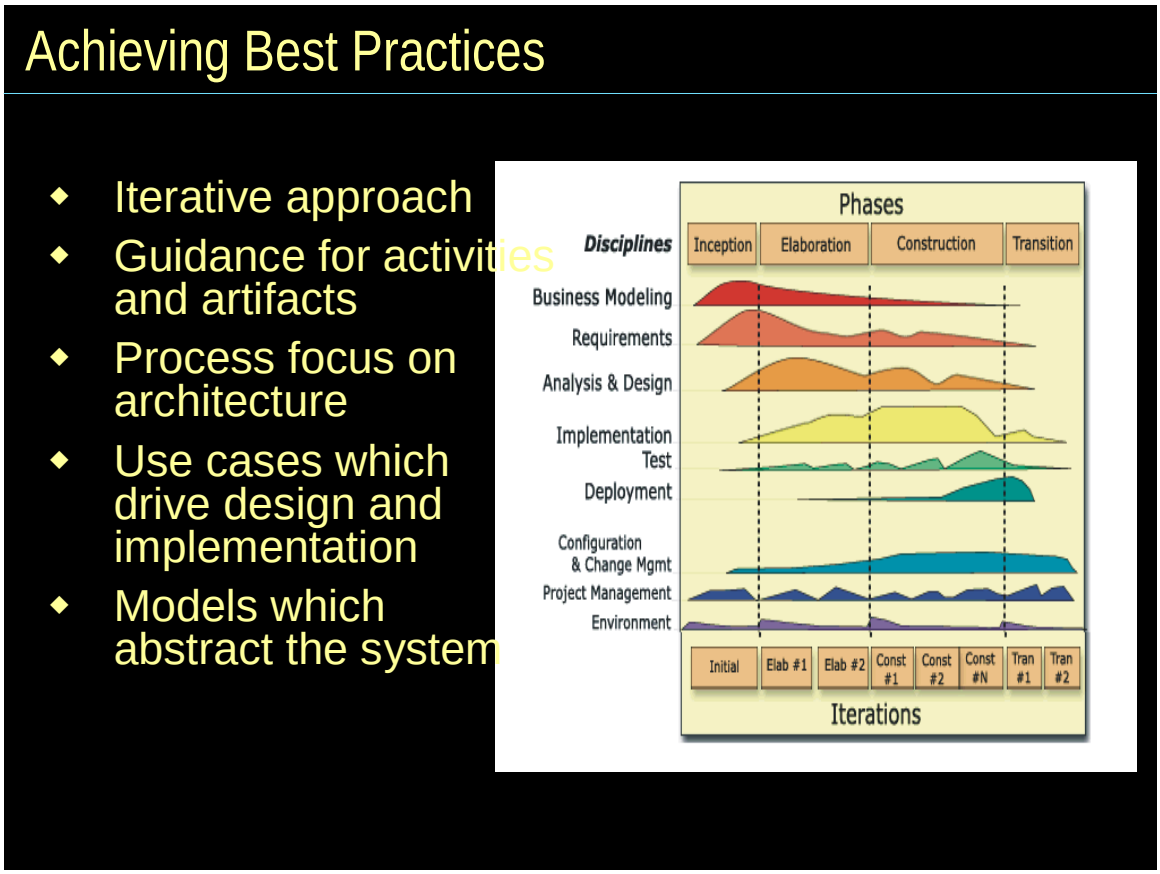


Bild: Der Softwareentwicklungsprozess Rational Unified Process (RUP)

## Process Structure - Lifecycle Phases


Rational Unified Process has four phases:

- **Inception** - Define the scope of project
- **Elaboration** - Plan project, specify features, baseline architecture
- **Construction** - Build the product
- **Transition** - Transition the product into end-user community

Bild: Phasen des Softwareentwicklungsprozess RUP

## SW-Entwicklung – Der Prozess

- **Beispiel: *Unified Software Development Process***
  **→ Ivar Jacobson, Grady Booch, James Rumbough**

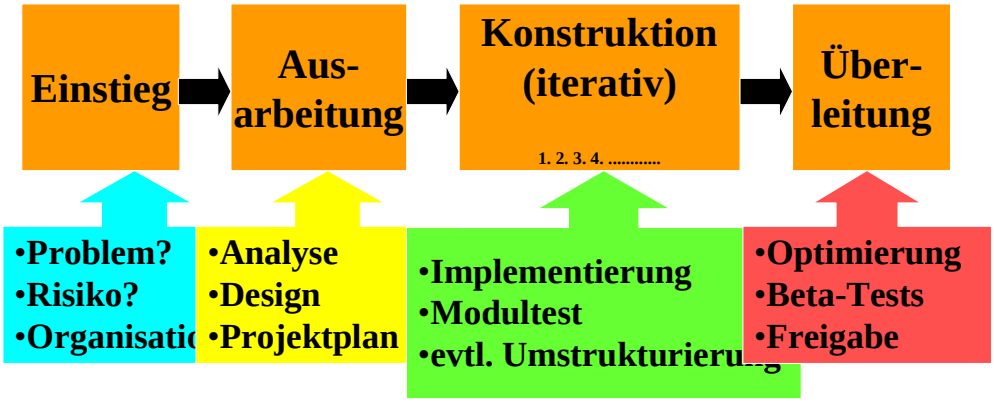


Bild: Phasen des Softwareentwicklungsprozess RUP

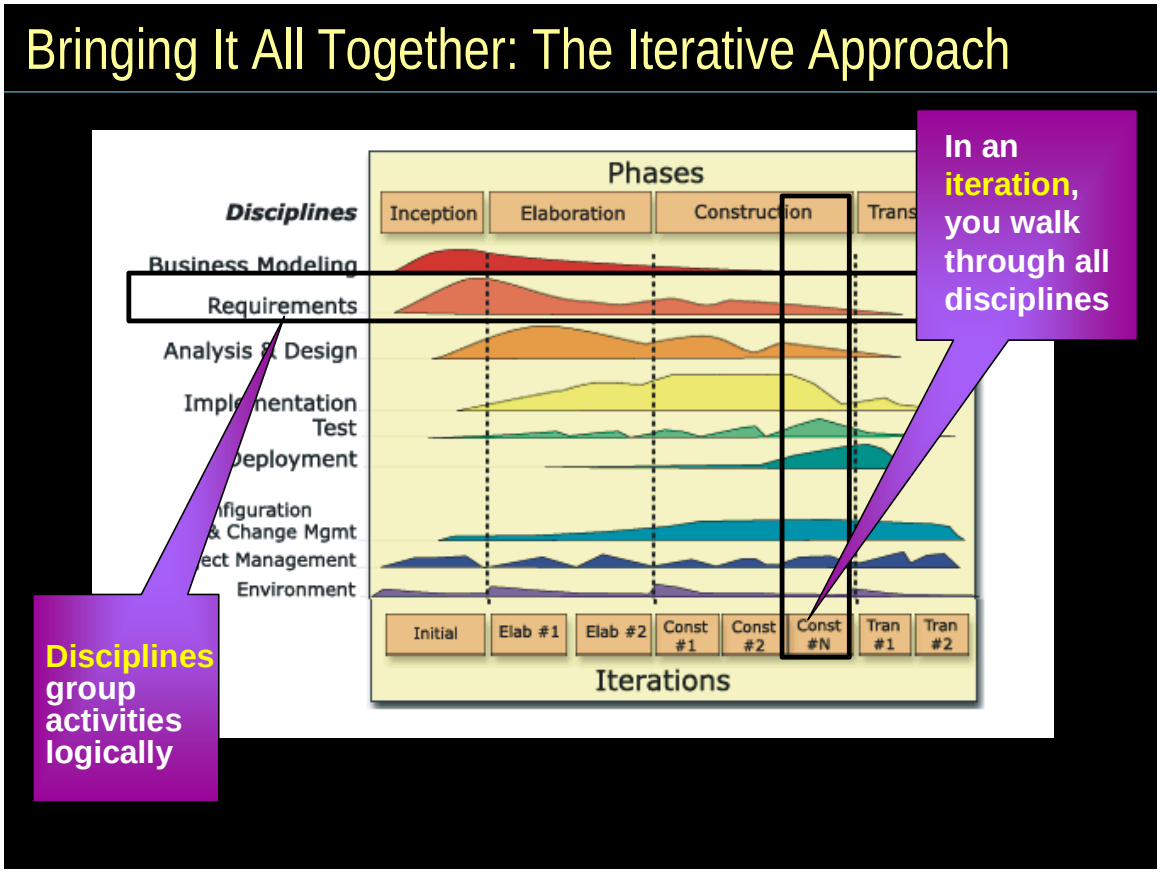


Bild: Iterationen des Softwareentwicklungsprozesses

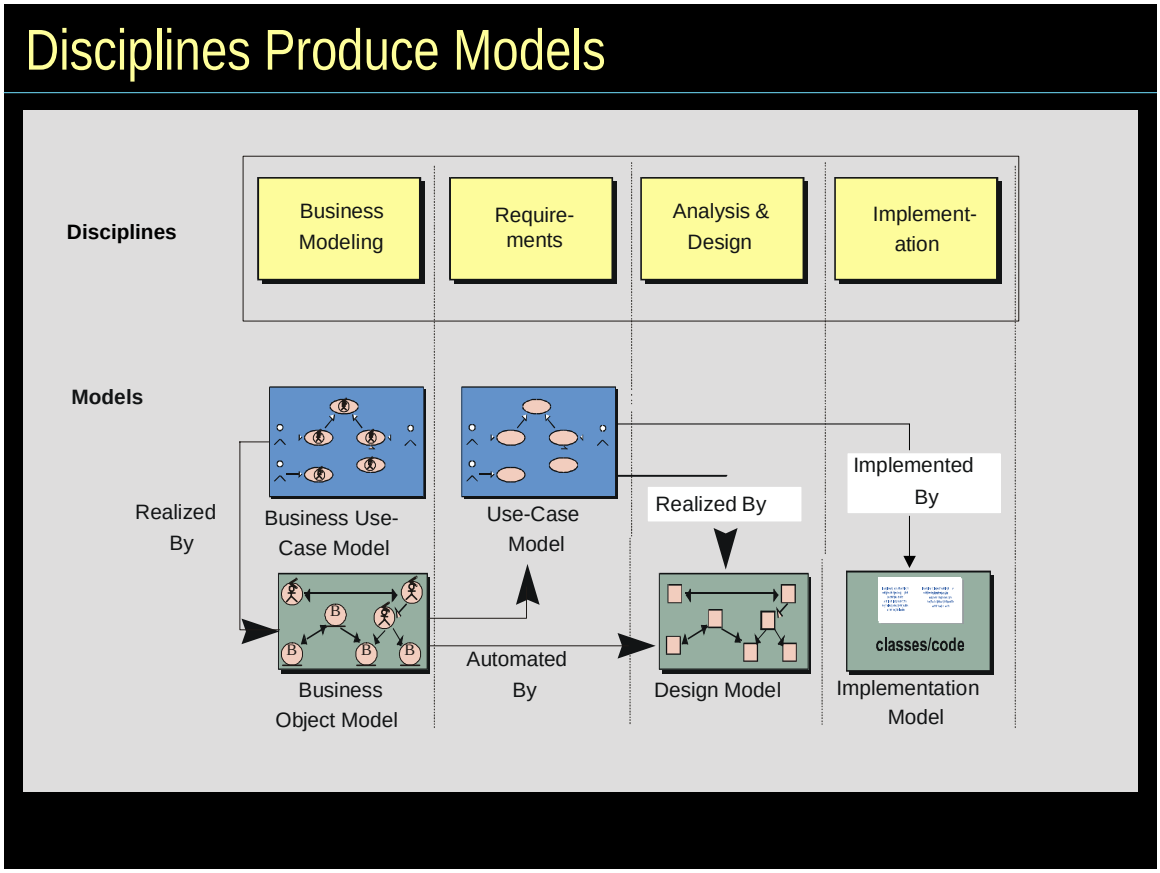


Bild: Generierte Dokumente