



Teil V: Projektmanagement

50 Projektplanung

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und
Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
TU Dresden
Version 21-0.2, 16.07.21

- 1) Projektmanagement
- 2) Vorgehensmodelle

Alles zu SalesPoint und zum Praktikum unter :

<http://st.inf.tu-dresden.de/SalesPoint/>

- ▶ Bestandteile der Dokumentation von SalesPoint sind:
 - Überblick und Einstieg
 - Technischer Überblick zu SalesPoint
 - Tutorial zu Anwendungsbeispielen (VideoShop, Guestbook)
 - API-Spezifikation der Framework-Klassen (javadoc) einschließlich der Beschreibung des Anpassungsinterfaces
- ▶ Dokumentation zahlreicher studentischer Praktikumsprojekte
- ▶ Infos zur Praktikumsdurchführung

Weiterführende Literatur

- ▶ [Baker03] F. Terry Baker. 2003. Chief programmer team. Encyclopedia of Computer Science. John Wiley and Sons Ltd., GBR, 209–210.
<https://dl.acm.org/doi/abs/10.5555/1074100.1074209>
- ▶ C. Alan Jennings: Robert's Rules for Dummies (For Dummies (Lifestyles Paperback))



50.1 Projektmanagement

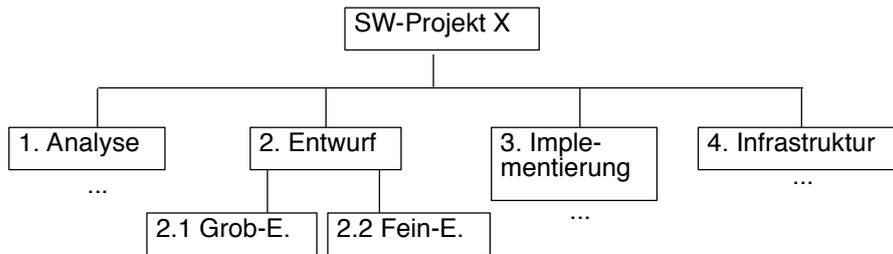
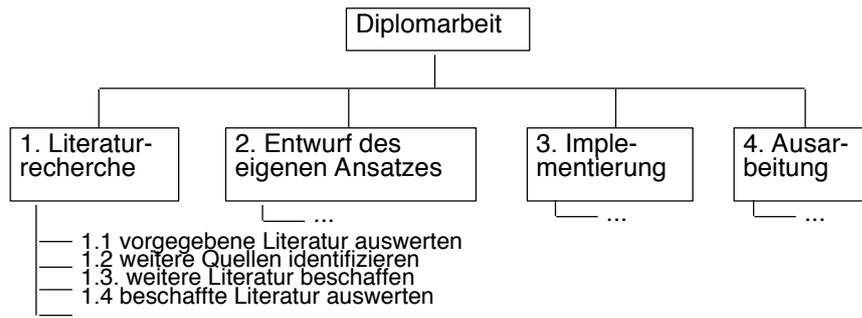
und eines
Projektes

Das Glück des Lebens besteht nicht darin, wenig oder keine Schwierigkeiten zu haben, sondern sie alle siegreich und glorreich zu überwinden.

Carl Hilty, 28.02.1833 - 12.10.1909

Schweizer Richter und Staatsrechtler, Buchautor und christl. Staatsrechts-Philosoph
Seine Bücher beeinflussten auch K. Adenauer

Projektstruktur ("Work Breakdown Structure"): Beispiele



Aufwandsschätzung

7 Softwaretechnologie (ST)

- ▶ Schätzungen für:
 - relativen Aufwand der Teilaufgaben
 - absoluten Aufwand für Subsysteme
- ▶ Faustregeln, Erfahrungswerte
- ▶ Techniken der Aufwandsschätzung:
 - Befragung von Entwicklern
 - Klassifikation z.B. durch "Function Point"-Methode
 - Wie viele Teilfunktionen?
 - Wie schwierig ist jede Teilfunktion?
 - Metriken für Spezifikationen
 - "Kalibrierung" durch eigene Erfahrungswerte

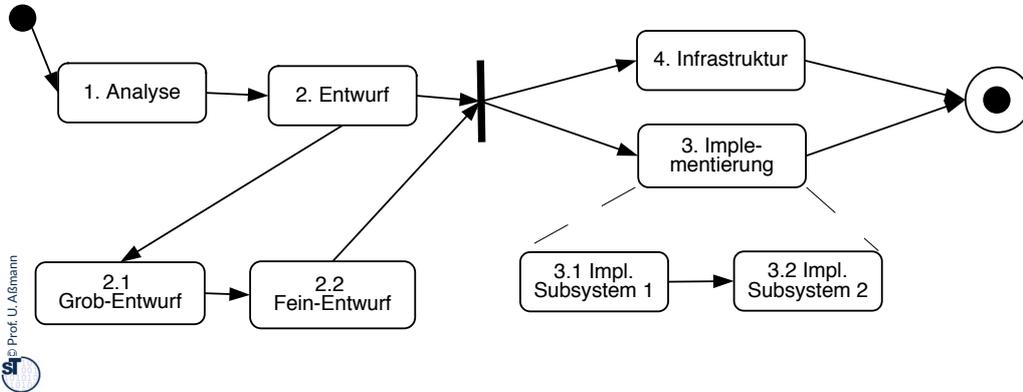


Mehr in Vorlesung „Softwaremanagement“, SoSe



Abhängigkeiten

- ▶ Welche Aktivitäten hängen von Ergebnissen anderer Aktivitäten ab? (Abhängigkeitsgraph)
- ▶ Aufwandsschätzung + feste Termine + Abhängigkeiten:
 - Netzplantechniken (z.B. PERT)
 - GANTT-Diagramm
- ▶ Beispiel für Abhängigkeiten, erfassbar in Aktivitätendiagramm:



Zeitplanung: Gantt-Diagramm, eine Aktivitätentabelle

9 Softwaretechnologie (ST)

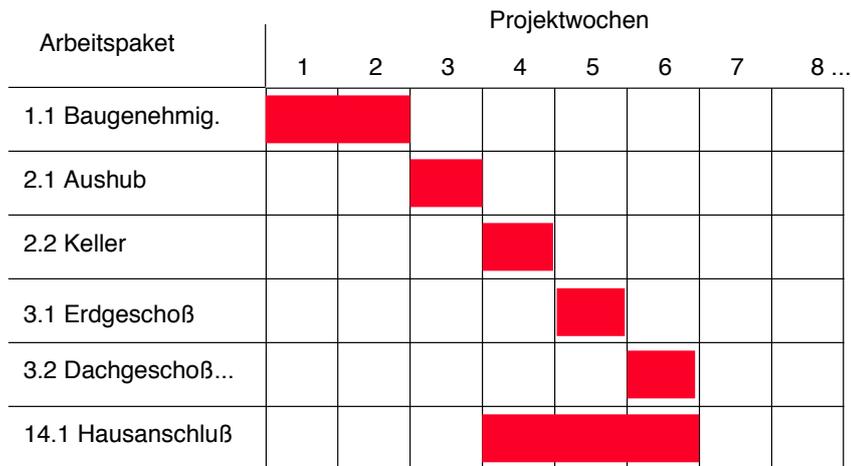
Arbeitspaket	Projektwochen							
	1	2	3	4	5	6	7	8 ...
1.1 Analyse	■	■						
2.1 Grobentwurf			■					
2.2 Feinentwurf				■	■			
3.1 Impl. Subsys. 1						■	■	
3.2 ff ...						■	■	■
4.1 Werkzeuge		■	■	■				

Identifikation *kritischer* und *unkritischer* (4.1, 3.1) Arbeitspakete
(kritisch = Verlängerung verlängert Gesamtprojektdauer)

Henry Gantt soll diesen Typ Diagramme im Verlauf des Ersten Weltkrieges entwickelt haben.

Zeitplanung Hausbau: Gantt-Diagramm

10 Softwaretechnologie (ST)

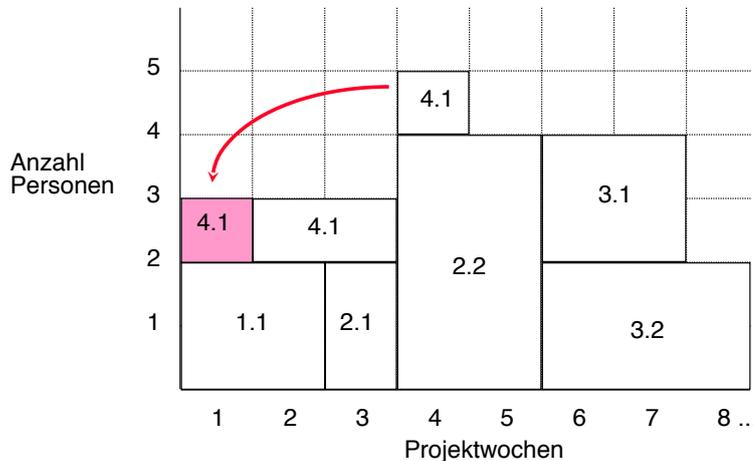


Henry Gantt soll diesen Typ Diagramme im Verlauf des Ersten Weltkrieges entwickelt haben.

Ressourcenplanung: Balancierung der Auslastung

11 Softwaretechnologie (ST)

- ▶ Umplanung mit dem Ziel: Anpassung an vorhandene Ressourcen
- ▶ Packen in Flächen über Anz. Personen und Projektwochen

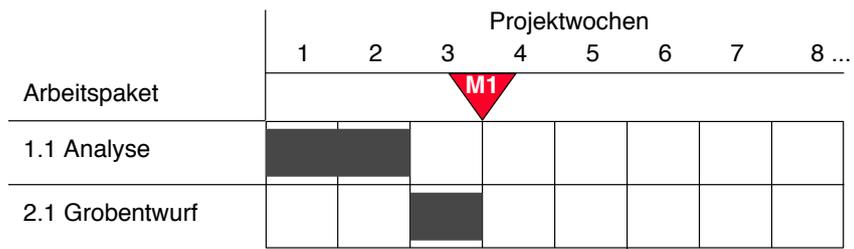


Eine naheliegende Optimierung ist, bereits in der 1. Woche mit 4.1 zu beginnen und somit für einen gleichmäßigeren Personalbedarf zu sorgen. Natürlich muß die Ressourcenplanung auch auf externe Einflüsse, z.B. andere Projekte, abgestimmt sein.

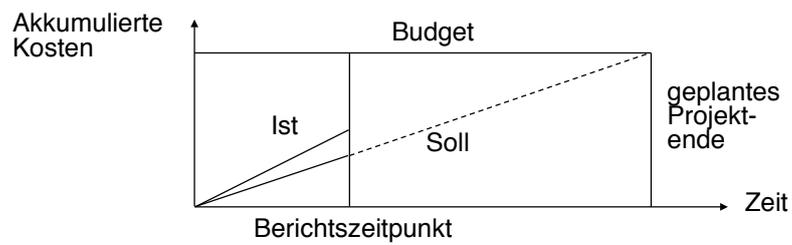
Ressourcen sind neben

Meilensteine

- ▶ Ein *Meilenstein* ist ein klar definiertes Zwischenresultat, an Hand dessen der Projektfortschritt beurteilt werden kann.
- ▶ Beispiele:
 - "Anforderungsspezifikation zusammen mit Auftraggeber verabschiedet"
 - "Erster Prototyp lauffähig"
 - Schlechtes Beispiel: "Code zu 50% fertig"
- ▶ Meilensteine im Gantt-Diagramm:

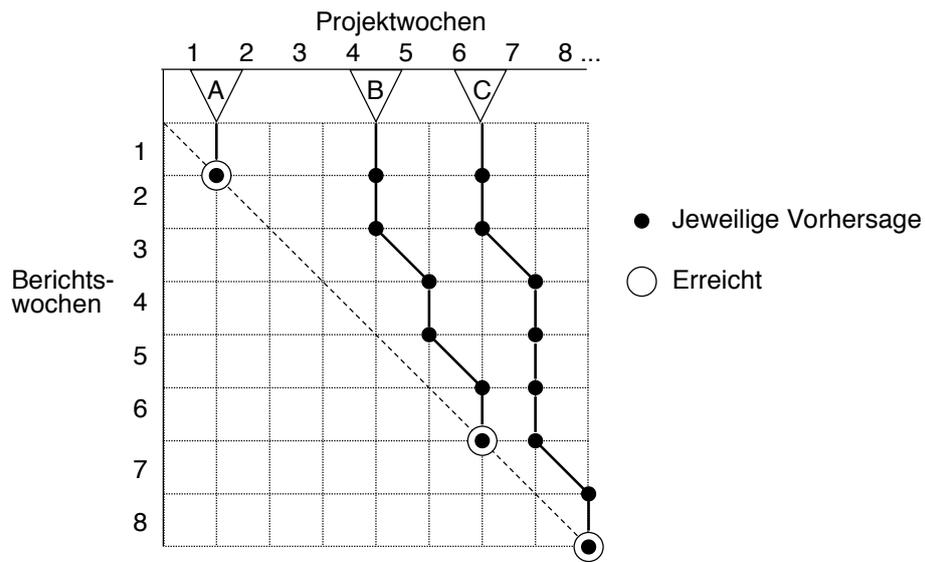


- ▶ Das Projektmanagement muß ein "Frühwarnsystem" für eventuelle Probleme betreiben (Projektverfolgung).
- ▶ Informationsquellen:
 - Laufende (z.B. wöchentliche) Management-Berichte
 - Arbeitszeit-Kontierung
 - Resultate (*deliverables*)
- ▶ Rückkopplung zum Projektteam
 - Regelmäßige Projektbesprechungen
 - Beispiel: Akkumulierter Ressourcenverbrauch



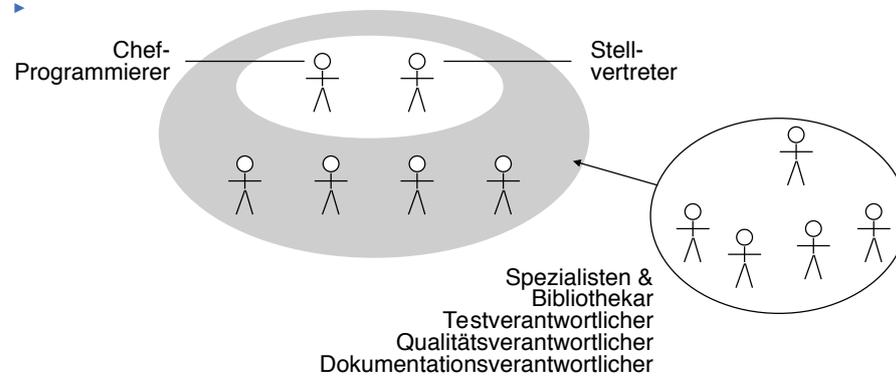
Meilenstein-Trendanalyse

- ▶ Anhand jedes Managementberichts sagt das Management die Meilensteine neu voraus



Teamzusammenstellung (Staffing)

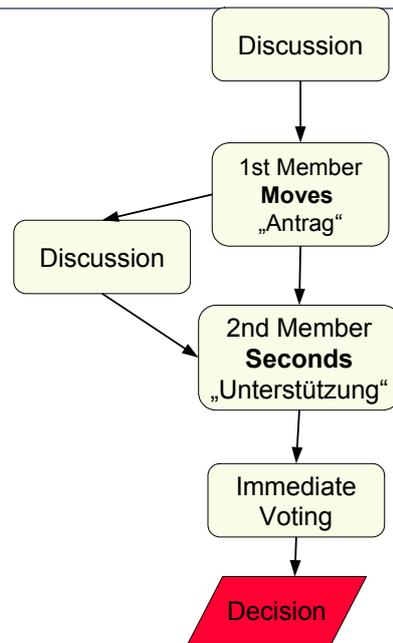
- ▶ Regeln für Teamproduktivität:
 - Optimale Teamgröße: ca. 5-7 Personen
 - Gemischte Qualifikationen
 - Team von externer Kommunikation entlastet
 - Große Projekte aus vielen Teams zusammengesetzt
- ▶ Harlan Mills / Baker 1972, [Baker03]: *Chefprogrammierer-Struktur*



Wie kommt man zu Entschlüssen? .. durch Robert's Rules of Order

17 Softwaretechnologie (ST)

- ▶ Demokratische Sitzungen sollten nach "Robert's Rules of Order for Debate" abgehalten werden
 - In USA als "parliamentary procedures" eingeführt, um ineffektive Sitzungen zu vermeiden
 - Jeder Amerikaner kennt sie, denn man lernt sie in der Schule...
- ▶ Definierte Schritte in der Sitzung.
 - "**Movement**" (Antrag zur Abstimmung) wird eingeleitet mit "I move for"
 - "**Secondment**" Antrag muss von zweitem Teilnehmer bestätigt werden "I second"
 - **Voting** Dann muss sofort über den Antrag abgestimmt werden. Wenn niemand unterstützt, entfällt der Abstimmungsantrag.
- ▶ Daneben gibt es etwa 10 weitere Regeln:
 - [Cummings]



Typische Gliederung eines Ergebnisprotokolls

- ▶ Name der Sitzung
- ▶ Teilnehmer, Moderator, Ort, Zeit
- ▶ Tagesordnung
 - Standard-Tagesordnungspunkte:
 - Protokollkontrolle
 - Bericht über den erreichten Stand
 - Einzelaufgaben
 - Nächster Termin
- ▶ Ergebnisse
 - gegliedert nach Tagesordnungspunkten (TOPs)
 - Beschlüsse
 - Ziele
 - Einzelaufgaben
 - Allokation von Einzelaufgaben an abarbeitende Personen
 - abgelehnte Anträge
 - vertagte Anträge



Was nicht in einem
Ergebnisprotokoll festgehalten
worden ist,
existiert nicht



Einzelaufgaben (*Action Items, Aktivitäten*)

- ▶ Einzelaufgabe (*action item, action point*) besteht aus:
 - Lfd. Nr., Verantwortliche Person
 - Kurztitel
 - Beschreibung
 - Ursprung (Sitzung, auf der Aufgabe definiert wurde)
 - Termin
 - Status (offen, verlängert, erledigt)
- ▶ Liste der Einzelaufgaben wird bei **jedem** Treffen durchgegangen und aktualisiert:
 - Welche Aufgaben sind fällig?
 - Was ist das Ergebnis?
 - Was ist weiter zu tun?
 - Termin verlängern
 - Neue Aufgaben definieren
- Können in einem *issue tracker* verwaltet werden (z.B. Mantis.org)



- ▶ **Einzelaufgaben (Aktivitäten)** werden in ein AufgabenmanagementäSystem eingetragen (“ticket system”, “issue management system”)
- ▶ Lesen Sie sich in den Semesterferien in eines ein, z.B. gitea

Werkzeug	Status	Webadresse
gitlab	community edition	Kanban boards of issues; https://gitlab.hrz.tu-chemnitz.de/ usable with ZIH account
gitea	free	try.gitea.io - free trial hosting
Bugzilla	Mozilla (OSS)	www.bugzilla.org
Mantis	OSS	http://www.mantisbt.org/
JIRA	Atlassian	http://atlassian.com/software/jira
codeBeamer	IntLand Software	http://intland.com/products/codebeamer/overview/
RedMine	OSS	http://en.wikipedia.org/wiki/Redmine
Team Foundation Server	Microsoft	http://en.wikipedia.org/wiki/Team_Foundation_Server

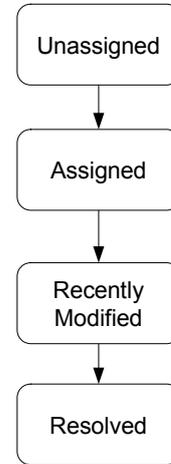
http://en.wikipedia.org/wiki/Comparison_of_issue_tracking_systems



Aufgabenmanagement (Fehler und Änderungen) mit Mantis

21 Softwaretechnologie (ST)

- ▶ Mantis ist ein webbasiertes Aufgabenmanagement-System (issue tracking system)
 - Zustandsmaschine für Fehler und Änderungswünsche
 - Generierung von emails über Statusänderungen
 - Visualisierung von Zuständen und Bearbeitern



Überblick über Aufgaben [www.mantisbt.org]

Viewing Issues (1 - 50 / 1520) | [Print Reports] | [CSV Export]

P	ID	US#	#	Category	Severity	Status	Updated	Summary
	0005069		3	GUI	minor	resolved (MacGyver)	2009-01-28	Invalid Password
	0005095			Other	minor	assigned (mow)	2009-01-28	error prueba
	0005091			Other	trivial	confirmed (ossgewalt)	2009-01-28	foo bar baz
	0004707			Other	minor	resolved (000willberty)	2009-01-28	relationships 1
	0005093			GUI	minor	assigned (enzyme)	2009-01-28	Test fichier attache
	0005082			Other	minor	assigned (121212)	2009-01-28	test
	0005094			GUI	minor	new	2009-01-28	Test fichier attache
	0005092			GUI	minor	new	2009-01-28	Test fichier attache
	0005090			GUI	minor	assigned (fandersen)	2009-01-28	Needs attention
	0005089		3	Website	major	resolved (Russell)	2009-01-28	Problems loading JavaScript on Main.html
	0005076		1	GUI	feature	assigned (darksaboteur)	2009-01-28	I can't poo...
	0005088			GUI	tweak	assigned (abarbosa)	2009-01-27	asdasdasda
	0005087			GUI	trivial	assigned (ramyap)	2009-01-27	asdasdasda
	0005086		9	GUI	trivial	assigned (Element)	2009-01-27	Detailbereich wird nicht angezeigt
	0005083		1	Other	minor	acknowledged (eaioc)	2009-01-27	test
	0005085			GUI	major	assigned (celso)	2009-01-27	teste com erro e associação com arquivo de repositório
	0005081			GUI	minor	resolved (deepak84)	2009-01-27	Sql Error
	0005080			GUI	feature	assigned (mmiat)	2009-01-27	????
	0005079			GUI	minor	assigned (kedar)	2009-01-26	test
	0005078		4	GUI	text	assigned (jodj)	2009-01-26	test
	0005077			GUI	minor	new	2009-01-26	can't abc
	0005069		3	GUI	tweak	new	2009-01-26	hish
	0005074		2	Other	minor	assigned (maxadmin)	2009-01-26	VIZ funktioniert nicht
	0005075			Website	major	assigned (patriciogomes)	2009-01-26	Erro no site
	0005071		2	Website	major	assigned (deepak84)	2009-01-26	Website Issue

Überblick über Zustände der Aufgaben

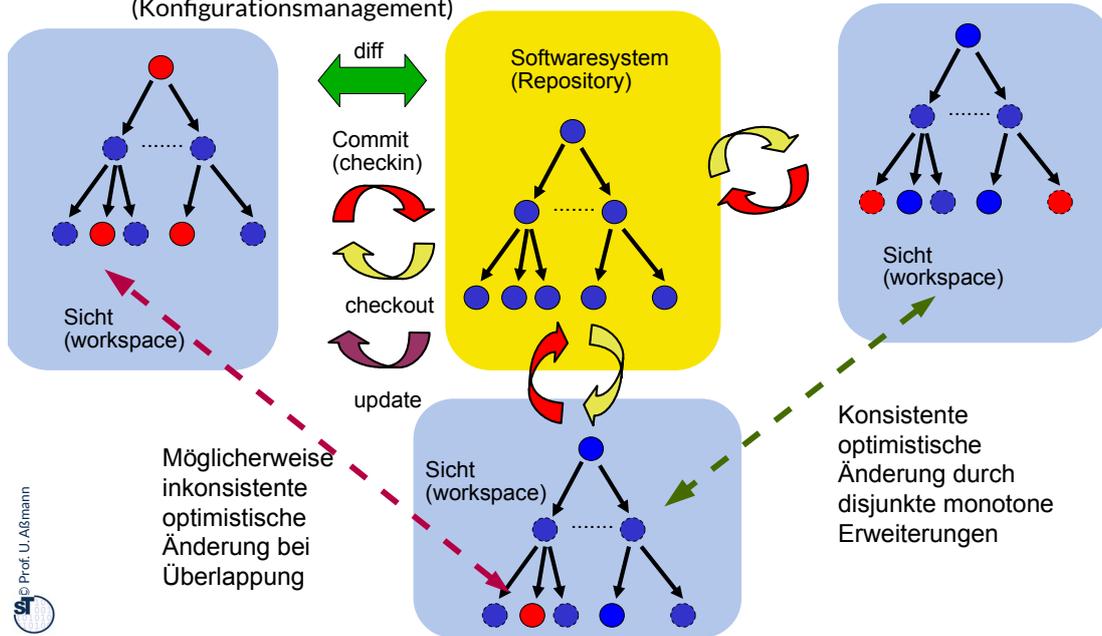
The screenshot displays the MantisBT web interface. At the top, there is a navigation bar with links for 'Main', 'My View', 'View Issues', 'Change Log', 'Roadmap', 'Docs', 'Wiki', and 'Billing'. The current date and time are shown as '2009-01-28 14:21 EST'. Below the navigation bar, there are three main panels:

- Unassigned [^] (1 - 10 / 271)**: A list of 10 unassigned tasks, each with a unique ID, description, and GUI version. The tasks are color-coded in red.
- Resolved [^] (1 - 10 / 154)**: A list of 10 resolved tasks, each with a unique ID, description, and GUI version. The tasks are color-coded in green.
- Recently Modified [^] (1 - 10 / 2234)**: A list of 10 recently modified tasks, each with a unique ID, description, and GUI version. The tasks are color-coded in various colors (blue, yellow, green, red, purple).

On the left side of the interface, there is a sidebar with a search bar and a list of navigation links. At the bottom left, there is a logo for 'Prof. U. Abmann'.

Zusammenarbeiten mit Konfigurationsmanagement

- Bei paralleler Bearbeitung müssen Sichten konsistent gehalten werden (Konfigurationsmanagement)



	Werkzeug		URL
Dateibaum- basiert	cvs	OSS	http://www.cvshome.org
	git	OSS	Linus Thorvalds, www.git-scm.com
Datenbank- basiert	ClearCase	IBM/Rational	http://www.rational.com/products
	Visual SourceSafe	Microsoft	http://www.eu.microsoft.com/ germany/produkte
beides	subversion	OSS	http://subversion.tigris.org
andere	Synergy	IBM Telelogic	http://www.telelogic.com/ product/synergy
	mercurial	Selenic, OSS	http://mercurial.selenic.com/
	InStep	microTOOL GmbH, Berlin	http://www.microTOOL.de

Konkrete Aufgaben zur Vorbereitung des Praktikums

- ▶ Einlesen:
- ▶ gitlab, gitea, mantis
- ▶ git oder subversion





50.2 Vorgehensmodelle (Phasenmodelle)

- ▶ Zuser Kap. 1-3 *oder*
- ▶ Ghezzi Chapter 1 *oder*
- ▶ Pfleeger Chapter 1; Chap 8.1

Vorgehensmodell (*engl. process model*)

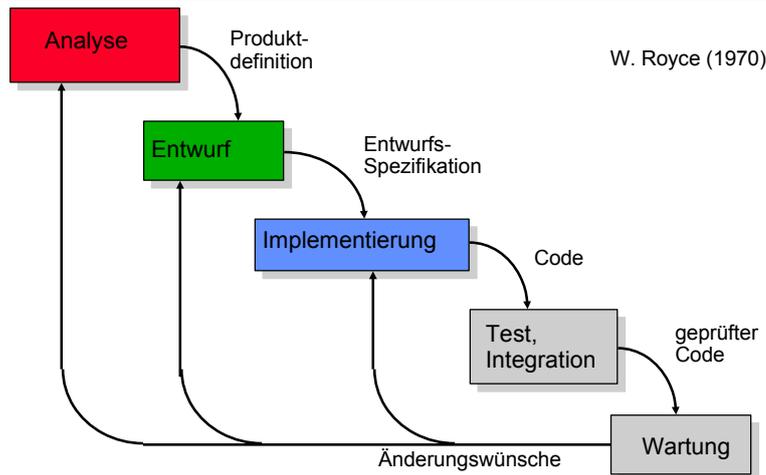
- Strukturiertes Modell zum Erstellen von Software

Phasenmodell

- Vorgehensmodell, das den Herstellungsprozesses in definierte und abgegrenzte Phasen einteilt
- Vorgabe einer Reihenfolge in der Bearbeitung der Phasen

Vorgehen nach einem “Phasenmodell”

- ▶ **Phasenmodell** (*process model, software development life cycle*)
 - Einteilung des Herstellungsprozesses für ein (Software-) Produkt in definierte und abgegrenzte Abschnitte, abgegrenzt durch **Meilensteine**
 - Grobgliederung: Phasen (*phases*)
 - Feingliederung: Schritte (*stages, steps*)
 - Vorgabe einer Reihenfolge in der Bearbeitung der Phasen
 - Richtlinie für die Definition von Zwischenergebnissen
 - Detailliertes Phasenmodell + Zwischenergebnisdefinition = „Vorgehensmodell“
- ▶ Grundaktivitäten:
 - Analyse
 - Entwurf
 - Implementierung
 - Validation (v.a. Test, Integration)
 - Evolution (v.a. Wartung)



W. Royce (1970)

- ▶ Das Wasserfallmodell ohne Iterationen ist nicht realistisch. Für ein Produkt müssen, schon um des Geschäftsmodells willen, Verbesserungen (Lebenszyklen) eingeplant werden
- ▶ Ein Lebenszyklus dauert i.D. 2 Jahre
- ▶ Dennoch muss ein Softwareingenieur den "Wasserfall" beherrschen, denn viele andere Vorgehensmodelle setzen darauf auf

Royce, 1970

Zu einem gewissen Grad ist die zyklische Anlage des Modells ein Eingeständnis der Tatsache, daß Software nicht im ersten Versuch in perfektem Zustand erstellt werden kann.

Die Realität ist nicht ganz so wie das Bild andeutet.

Phasen sind nicht immer klar voneinander zu trennen (z.B. unterschiedlicher Fortschritt in verschiedenen Systemteilen)

Lokale Rückkopplungen (z.B. vom Design zur Anforderungsdefinition) sind möglich und durchaus erwünscht (wenn sie größere/spätere Zyklen vermeiden).

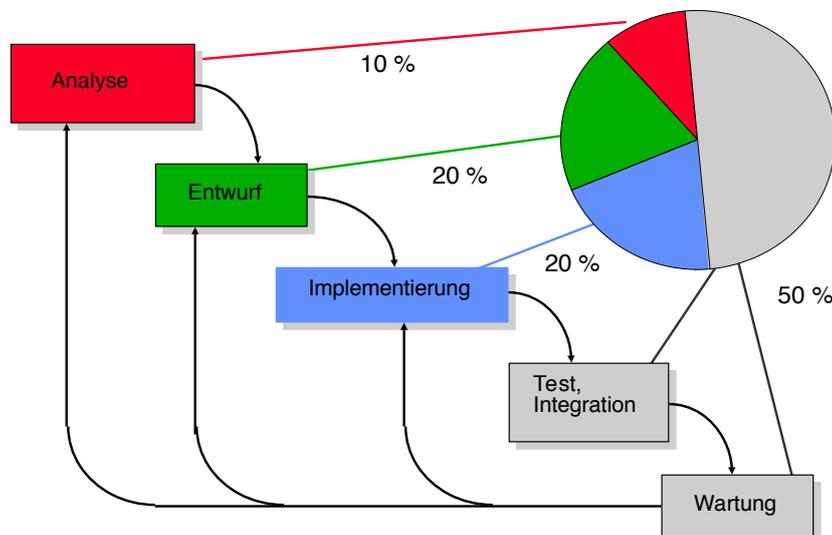
Kritik am Wasserfall-Modell:

- Implizite Annahme, dass sich die Anforderungen während des Projektverlaufs nicht ändern (no-change assumption)
- Das Ende einer Phase wird als perfekte Teillösung aufgefasst (perfect understanding assumption)
- Benutzer sehen das System erst, wenn es fertig ist
- Reines top-down-Vorgehen ist nicht zielführend: Bestimmte Probleme der Implementierung z.B. müssen früh erkannt werden und können sogar die Anforderungsdefinition beeinflussen.

Vorteil des Modells: Abschirmung gegen ständige

Ungefähre Verteilung des Arbeitsaufwandes

32 Softwaretechnologie (ST)



In der industriellen Praxis spielt das Wasserfallmodell noch eine sehr wichtige Rolle, und zwar vor allem als Orientierungsrahmen, der mehr oder minder modifiziert wird.

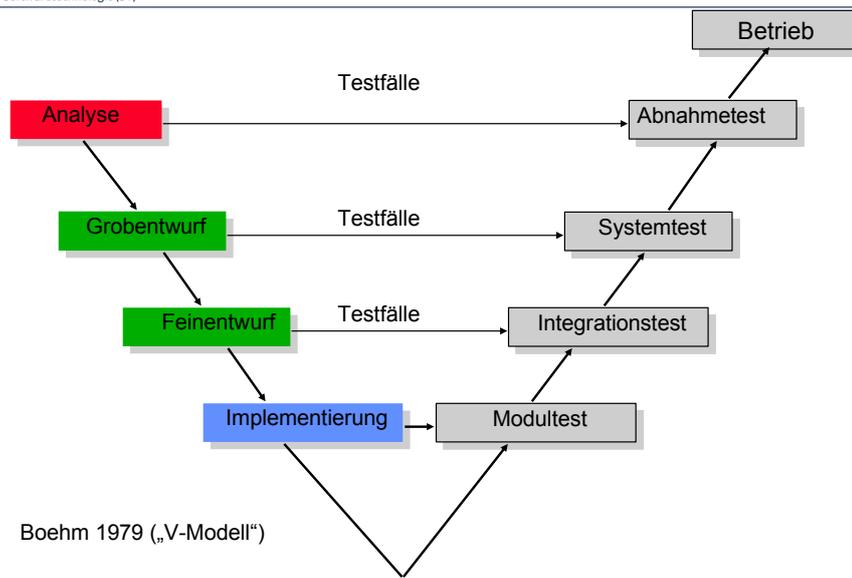
Z.B. dient die Phasenidentifikation als gute Basis für Faustregeln zur Abschätzung der Aufwandsverteilung.

Der Arbeitsalltag in der Softwareentwicklung enthält noch weitere Tätigkeiten, die den Anteil der reinen Programmierfähigkeit noch weiter reduzieren.

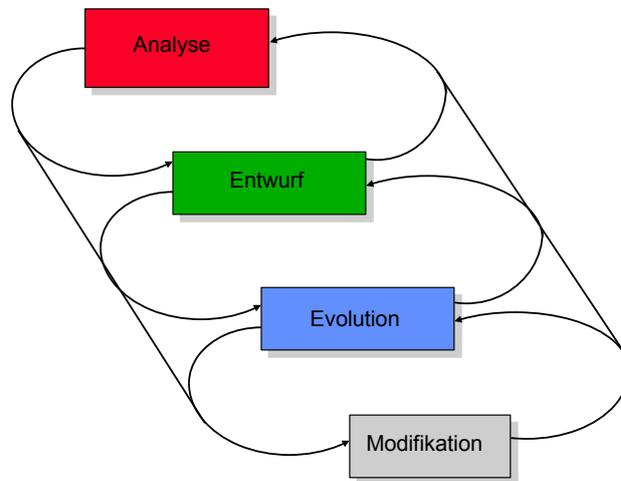
Z.B. gaben Softwareentwickler bei einer Umfrage an, sie verwendeten ca. 15% ihrer Arbeitszeit auf das Schreiben von Programmen, während z.B. 30% auf die Kommunikation mit Kollegen verwendet werden.

Qualitätssicherung im einfachen V-Modell

33 Softwaretechnologie (ST)



Quelle fuer das Boehm-Zitat: Hesse/Merbeth/Frölich
S. 37



Dieses Modell basiert auf Vorschlägen von G. Booch (ca. 1992). Darstellung nach Bannert/Weitzel 1999.

Inkrementelle Entwicklung kann mit jeder Programmier- und Modellierungssprache durchgeführt werden. Besondere Bedeutung erlangte diese Vorgehensweise allerdings im Zusammenhang mit objektorientierter Systementwicklung, wo sie von frühen Autoren bewußt als Gegensatz zur traditionellen Wasserfall-artigen Entwicklung propagiert wurde. Hier wird bewußt eine Vermischung aller Aktivitäten der Entwicklung angestrebt.

Probleme mit inkrementeller Entwicklung:

- Ungeplante Systemstruktur

- Mangelhafte Dokumentation

- Fortschritt nicht kontrollierbar

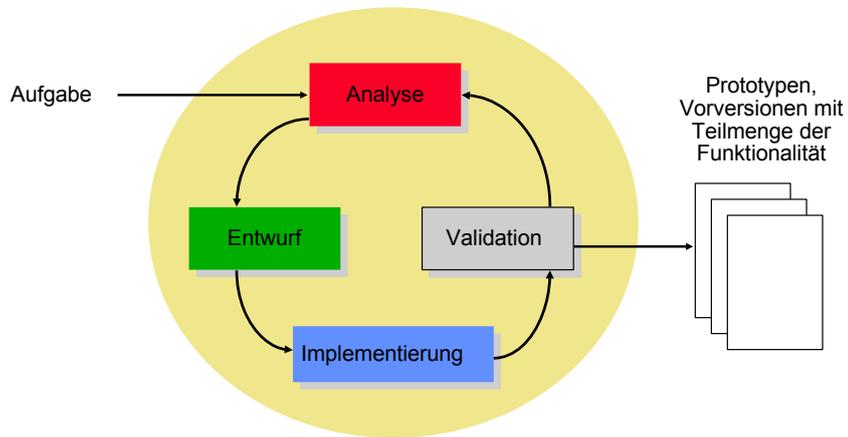
- Schwierig zu schulen

- Problematisch für das Management großer Projekte

Evolutionäre Entwicklung

36 Softwaretechnologie (ST)

- ▶ Typisch für kleinere Projekte oder experimentelle Systeme
- ▶ Bei Objektorientierung auch für größere Projekte anwendbar ?



Probleme mit rein evolutionärer Entwicklung:

Schlechte Systemstruktur

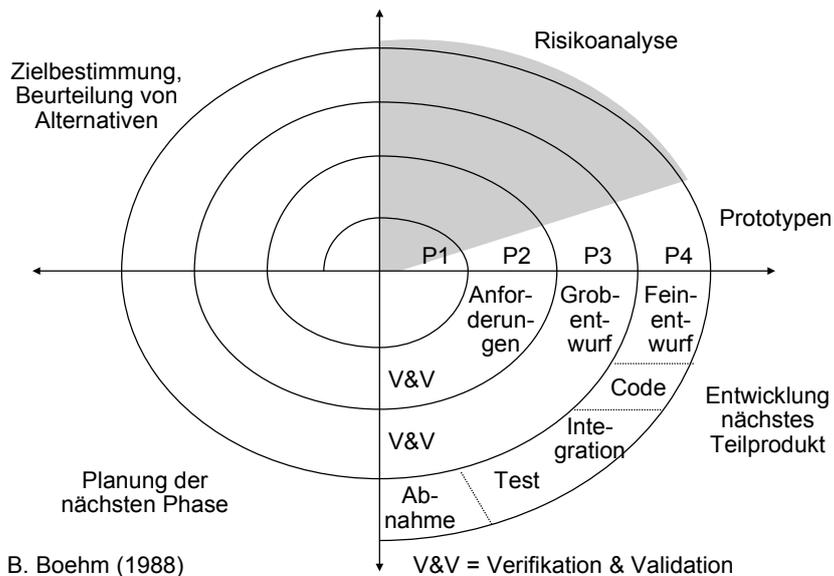
Schlechte Dokumentation

Vorgehensmodell nicht wahrnehmbar

Schwierig zu schulen

eXtreme Programming (XP)

- ▶ Kontrovers diskutierte Entwicklungsmethodik (Kent Beck)
 - Konsequente evolutionäre Entwicklung
 - Der Programmcode ist das Analyseergebnis, das Entwurfsdokument und die Dokumentation. Code wird permanent (Tagesrhythmus) lauffähig gehalten
 - Diszipliniertes und automatisiertes Testen als Qualitätssicherung
 - Diverse weitere innovative Techniken (z.B. Paar-Programmierung)
 - liefert schnell Ergebnisse, aber u.U. auf Kosten der Langlebigkeit
 - kann prinzipiell mit traditionelleren Analyse- und Entwurfstechniken kombiniert werden
- ▶ Nachteile
 - wird manchmal als Gegenbewegung zu sauberem Softwareentwurf **miß**verstanden
 - ist nur geeignet für relativ überschaubare, isolierte Anwendungen
- ▶ "Agile" Softwareentwicklung (www.agilemanifesto.org):
 - weitere Ansätze, z.B. Crystal, Scrum



Versuch eines Kompromisses zwischen Wasserfallmodell und evolutionärem Vorgehen.

Regelmässige Zwischenergebnisse in der Form von **Prototypen**.

Wesentlich: Betonung der Risikoanalyse und Planung der Folgephasen.

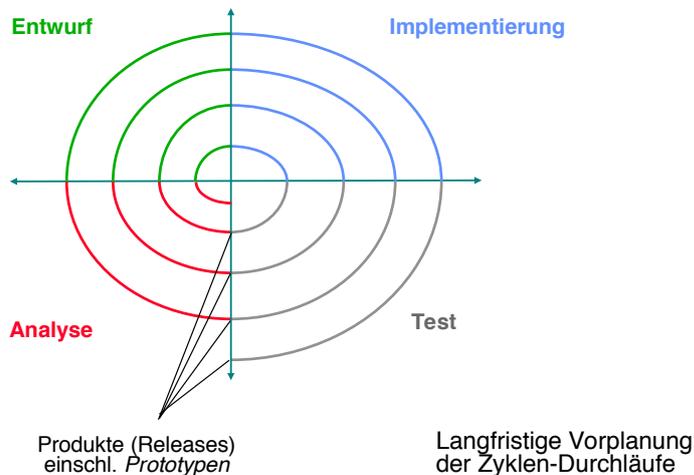
Wurde populär im Zusammenhang mit den sogenannten Programmiersprachen der 4. Generation (4GL) (also noch vor der Popularitätswelle der Objektorientierung).

Neuere Versionen (1994-1998): Betonung der Identifikation sogenannter 'stakeholder' (Beteiligter) und der Findung von 'win-conditions' (Erfolgsbedingungen), insbesondere solcher, die allen Beteiligten Vorteile verschaffen (Aushandlung von 'win-win conditions'). Diese Theorie ist bekannt unter dem Namen 'Theory W' und das betreffende Spiralmodell wird 'WinWin-Spiralmodell' genannt.

(Quelle: B. Boehm et al., IEEE Computer, July 1998, pp. 33-44)

Kritik am Spiralmodell (nach S. Ambler, Process Patterns):

- nicht realistisch - es werden oft Zwischenphasen übersprungen, z.B.: wenn Probleme entdeckt wurden, geht man oft direkt zur Zielbestimmung/Risikoanalyse über
- zu komplex für die praktische Anwendung



Ein Versuch, das zyklische Vorgehen mit einem Fortschritt im Sinne des Wasserfallmodells zu verbinden, ist das sogenannte „Spiralmodell“ von B. Boehm 1988, das allerdings eine etwas andere Gliederung der Durchläufe vorsieht.

Wesentlich ist, daß nach einem Durchlauf durch den Zyklus ein Zwischenergebnis vorliegt, das als Meilenstein der Projektplanung dienen kann. Häufig sind solche Zwischenergebnisse Prototypen. Prototypen können weiterentwickelt, aber auch weggeworfen und komplett ersetzt werden.

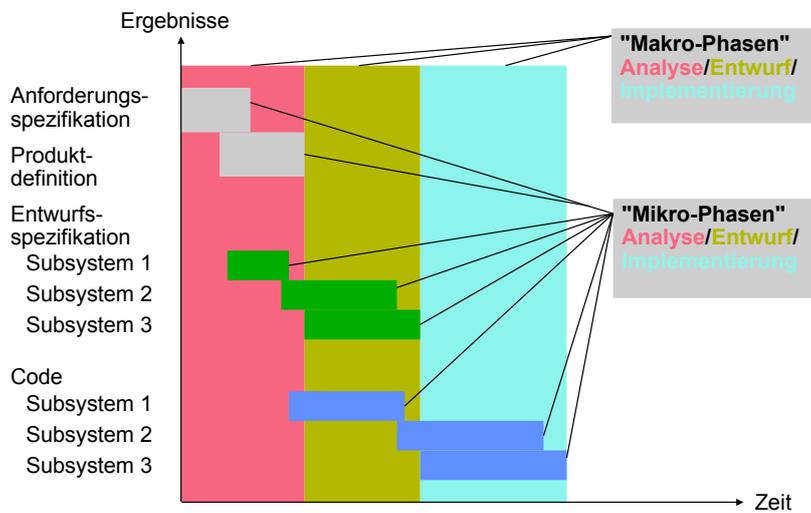
Probleme mit einer solchen Variante des Spiralmodells:

- Die Darstellung, daß auch in frühen Stadien des Projekts z.B. eine Implementierungstätigkeit stattfindet, ist etwas irreführend. Erste Zwischenergebnisse können auch z.B. reine Textdokumente sein.
- Der Gesamtfortschritt des Projekts wird nicht mehr sichtbar, außer durch die Zwischenergebnisse.

- ▶ Grundidee identisch:
 - Zyklisches Durchlaufen von Entwicklungsaktivitäten
 - Aufeinanderfolgende Prototypen
- ▶ Evolutionäre und agile Entwicklung:
 - Reaktion auf Änderungen ist wichtiger als Verfolgung eines Plans
 - Planung nur für sehr kurze Zeiträume (Tage, Wochen) im voraus
 - Viele, häufige Durchläufe (z.B. Tagesrhythmus)
- ▶ Spiralmodell:
 - Einsetzbar in verschiedener "Strenge"
 - Vorausplanung von Durchläufen
 - Anzahl Durchläufe manchmal schon bei Projektbeginn festgelegt
 - Wenige Durchläufe (z.B. Quartalsrhythmus)
 - Kompromiß zwischen Planbarkeit und Agilität

Parallelität im Entwicklungsprozeß

41 Softwaretechnologie (ST)



Hier wird betont, daß einzelne Systemkomponenten und auch andere Produkte der Entwicklung ihren eigenen Lebenszyklus haben. Der Gesamtverlauf des Projekts ist eine Komposition aus solchen "Mikro-Lebensläufen".

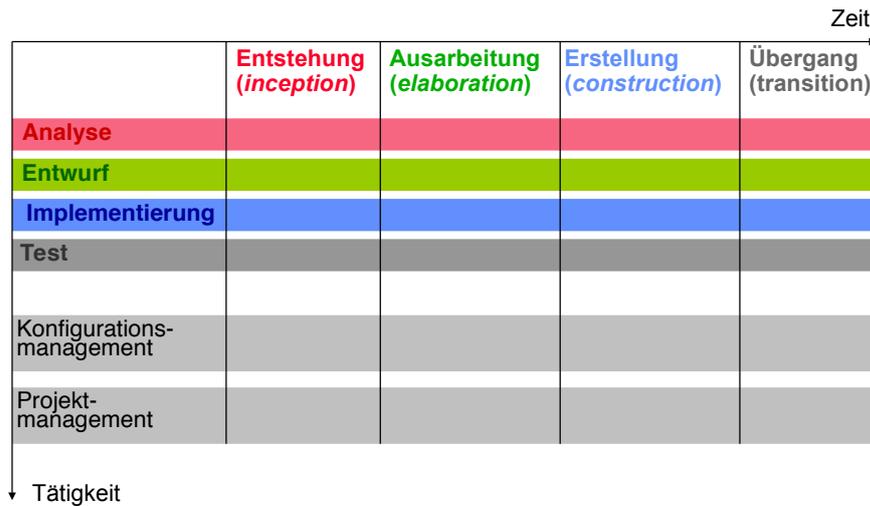
Neben echten Subsystemen haben z.B. auch experimentelle und explorative Prototypen einen eigenen „Mikro-Lebenslauf“. Generell kann man sagen, daß in der Praxis die Makrophasen durchaus Tätigkeiten „fremder“ Natur enthalten, z.B. Entwurfs- oder sogar Wartungsüberlegungen in der Analysephase.

Die „Makrophasen“ tragen nun keine sehr passenden Namen mehr, da es ja durchaus vorkommt, daß z.B. in der Makrophase „Analyse“ bereits Entwurfs- und Implementierungstätigkeiten stattfinden.

Diese pragmatische Beobachtung beweist, daß sogar beim Versuch, ein rein traditionelles Wasserfallmodell anzuwenden, zusätzliche Komplexitäten auftreten, insbesondere ein zweidimensionaler Charakter des Prozesses. Für weitere zweidimensionale Modelle siehe weiter unten.

Zweidimensionales Modell

- ▶ Rational Unified Process 1999 (Jacobson et al., Kruchten) mit Mikro- und Makrophasen

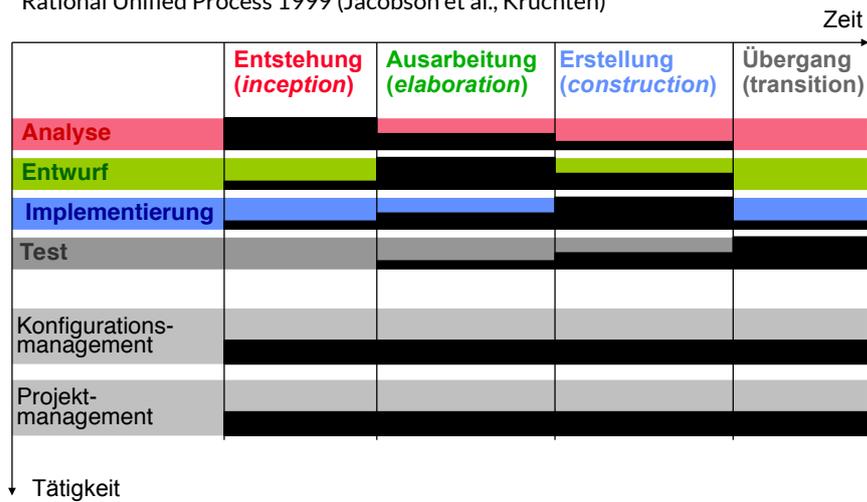


Es ist eine wesentliche Erkenntnis moderner Ansätze, wie des Rational Unified Process, daß zwei Dimensionen der Entwicklung existieren. Zu einem gewissen Grade kann die Entwicklung in der Zeit unabhängig von den auftretenden Aktivitäten angesehen werden, weshalb es sinnvoll ist, neue Namen für die klassischen Projektphasen (in zeitlichen Sinne) zu verwenden.

Das klassische Wasserfallmodell identifiziert Analyse/Entstehung, Entwurf/Ausarbeitung, Implementierung/Erstellung und Test/Übergang.

Das objektorientierte Spiralmodell erscheint hier als eine "schraubenartige" Bewegung in der aufgespannten Fläche.

Rational Unified Process 1999 (Jacobson et al., Kruchten)



Dies ist eine typische Aufwandsverteilung, die wesentliche Ähnlichkeiten mit dem Wasserfallmodell zeigt.

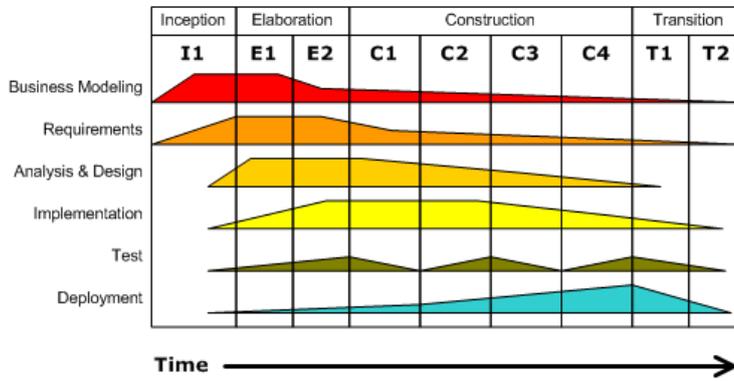
Bei anderer Aufwandsverteilung sind andere Modelle problemlos integrierbar, z.B. ein Extreme-Programming-Modell, bei dem im wesentlichen alle Tätigkeiten in etwa über die Entwicklung gleich stark ausgeprägt werden, mit einer relativen Betonung der Test-Tätigkeit.

Rational Unified Process (RUP)

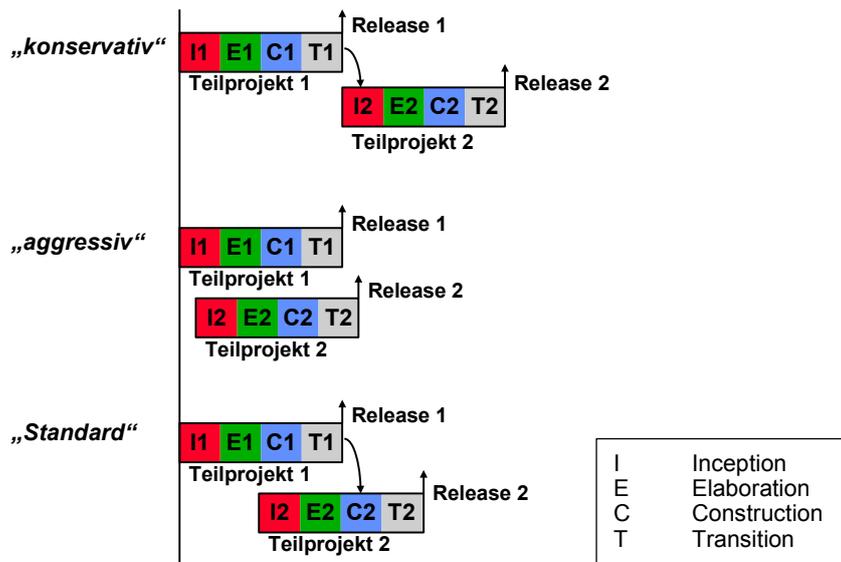
- ▶ von IBM Rational. INECT Process in iterations (INception, Elaboration, Construction, Transition):

Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.



<https://de.wikipedia.org/wiki/Datei:Development-iterative.png>



Diese Überlappungsgrade werden von S. Ambler im Buch „Process Patterns“, S. 41, diskutiert, wo eine ähnliche Gliederung des zeitlichen Verlaufs (Initiate, Construct, Deliver, Maintain and Support) eingeführt wird.

Die "Inception"-Phase (bzw. "Initiate") wird bei späteren Releases meist wesentlich schwächer ausgeprägt sein als beim ersten Release.

Der "konservative" Ansatz ist relativ einfach zu überschauen, aber in der Praxis durch kurzfristige Kundenanforderungen und Markterfordernisse oft zu schwerfällig. Der "aggressive" Ansatz verspricht auf den ersten Blick kurze Folge von releases, wirft jedoch ein enormes Problem der Versionsverwaltung auf, da gleichzeitig an zwei Versionen desselben Codes bzw. der selben Spezifikation gearbeitet wird. Es ist wesentlich sicherer, eine Überlappung gleichartiger Phasen zu vermeiden, wie z.B. in der "Standard"-Variante oben angedeutet.

Vorgehen im Softwarepraktikum 3. Semester

- ▶ Echte Kunden
- ▶ Vorgehensmodell: V-Modell mit Akzeptanztests
- ▶ Einfache Inkrementalität: Kunde hat einen *Verbesserungswunsch* frei, der erst zu einem späten Zeitpunkt bekanntgegeben wird
- ▶ Intern kann ein inkrementelle Vorgehensmodell gewählt werden

Was haben wir gelernt?

- ▶ Vorgehen nach einem strukturierten Phasenmodell ist gewöhnlich besser als ad-hoc Vorgehen
- ▶ Realistische Vorgehensmodelle sind iterativ und inkrementell
- ▶ Der Ingenieur misst, entwirft, validiert und verbessert

Thus it will be seen that *engineering is a distinctive and important profession*. To some even it is the topmost of all professions. However true that may or may not be to-day, certain it is that some day it will be true, for the reason that engineers serve humanity at every practical turn.

Engineers make life easier to live--easier in the living; their work is strictly constructive, sharply exact; the results positive.

Charles M. Horton. Opportunities in Engineering. 1920, by Harper & Brothers
<http://www.gutenberg.org/ebooks/24681>



.. Not a profession outside of the engineering profession but that has its moments of wabbling and indecision--of faltering on the part of practitioners between the true and the untrue. Engineering knows no such weakness. *Two and two make four. Engineers know that.* Knowing it, and knowing also the unnumbered possible manifoldings of this fundamental truism, engineers can, and do, approach a problem with a certainty of conviction and a confidence in the powers of their working-tools nowhere permitted men outside the profession.

Charles M. Horton. Opportunities in Engineering. 1920, by Harper & Brothers

<http://www.gutenberg.org/ebooks/24681>



- ▶ Die deutschen Folien der Softwaretechnologie-Vorlesung stammen zu Teilen von Prof. Dr. Heinrich Hussmann. Used by permission.