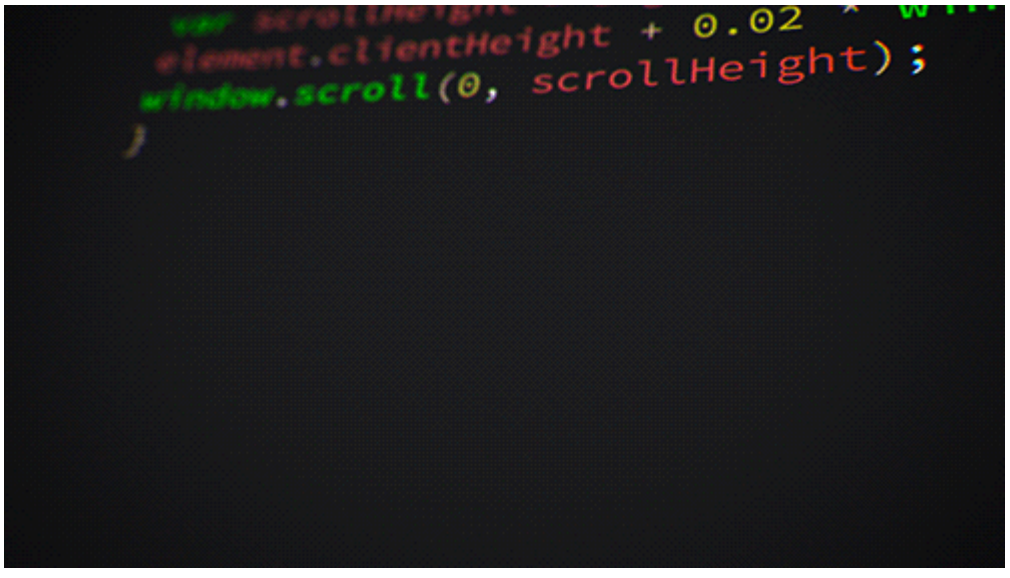
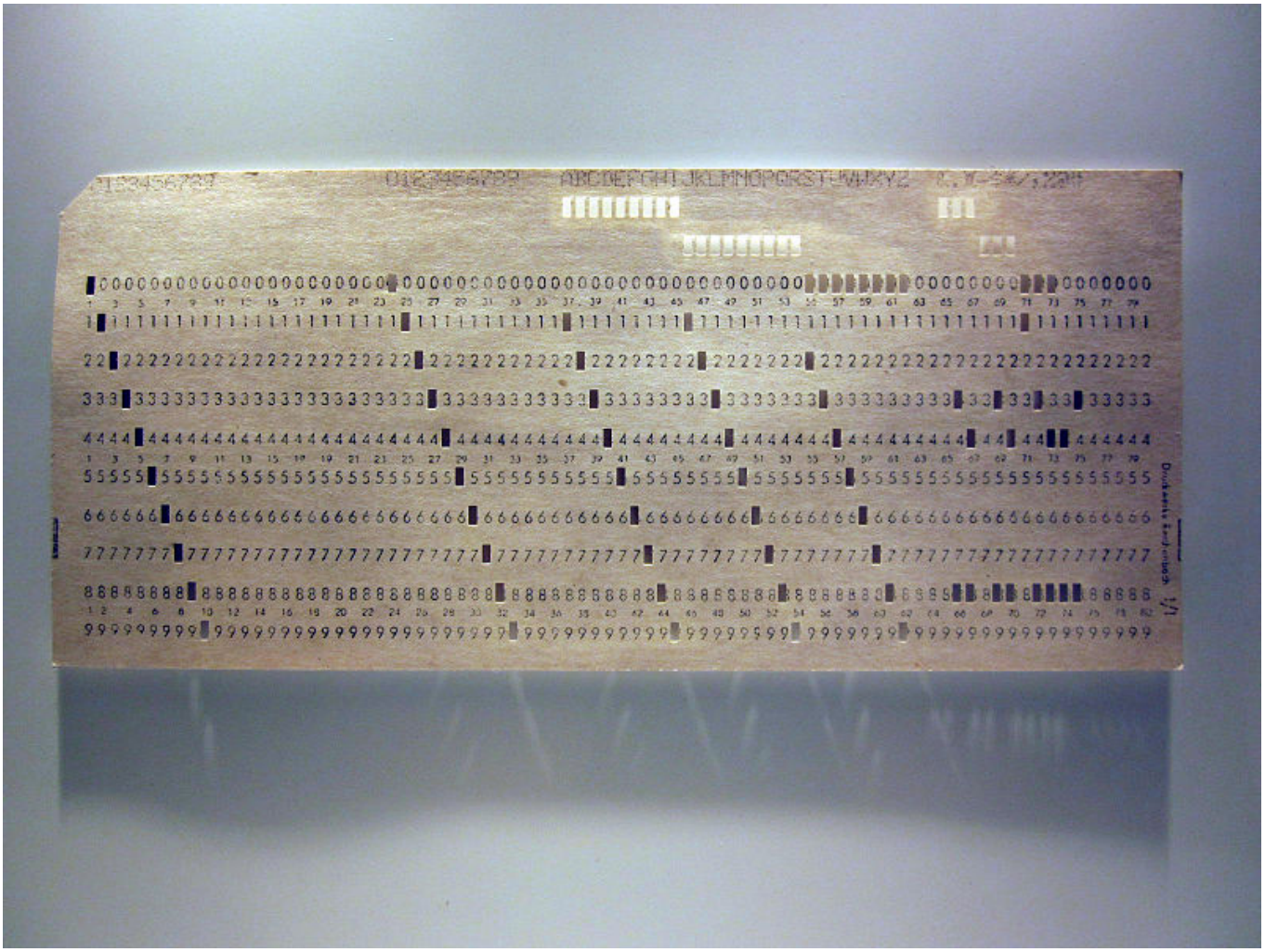
Softwareentwicklung als Prozess

Parameter	Kursinformationen
Veranstaltung:	`Vorlesung Softwareentwicklung`
Teil:	`1/27`
Semester	`Sommersemester 2025`
Hochschule:	`Technische Universität Freiberg`
Inhalte:	Definition des Softwarebegriffes, Softwareentwicklungszyklus, Fehler in der Softwareentwicklung, Softwarequalität
Link auf den GitHub:	https://github.com/TUBAF-IfI-LiaScript/VL_Softwareentwicklung/blob/master/01_Software.md
Autoren	Sebastian Zug; Galina Rudolf; André Dietrich; Christoph Pooch; `KoKoKotlin`; `Lina` & `Florian2501`



Softwareentwicklung

Was ist Software, welche Abläufen kennzeichnen den zugehörigen Entwicklungsprozess?



Denis Apel, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons

```
00000000  48 69 65 72 20 69 73 74  20 65 69 6e 20 42 65 69  |Hier ist ein B
00000010  73 70 69 65 6c 74 65 78  74 2e 20 44 65 72 20 48  |spieltext. Der
00000020  65 78 64 75 6d 70 20 69  73 74 20 61 75 66 20 64  |exdump ist auf
00000030  65 72 20 6c 69 6e 6b 65  6e 20 53 65 69 74 65 20  |er linken Seit
00000040  7a 75 20 73 65 68 65 6e  2e 0a 0a 4e 65 75 65 20  |zu sehen...Neu
00000050  5a 65 69 6c 65 6e 20 6f  64 65 72 20 41 62 73 e4  |Zeilen oder Ab
00000060  74 7a 65 20 73 69 6e 64  20 64 61 6e 6e 20 61 75  |tze sind dann
00000070  63 68 20 22 5a 65 69 63  68 65 6e 22 20 6d 69 74  |ch "Zeichen" m
00000080  20 65 69 6e 65 6d 20 62  65 73 74 69 6d 6d 74 65  | einem bestimm
00000090  6e 0a 43 6f 64 65 2e 28  30 61 29 2e 2e 2e 0a 0a  |n.Code.(0a)...
```

Begriffsdefinition

Begriff	Definitionsansatz
Software als "Medium"	*Software [ist] all das, was zum Funktionieren eines Computers notwendig, aber nicht Hardware ist.*
	Software ist sinnlich nicht wahrnehmbar ... . Sie ist komplex und besteht aus umfangreichen Texten
Software als Ziel	*Software macht den Computer nutzbar*
	Software ermöglicht die Abbildung von Prozessen auf einem Rechner
Software als Ganzes	*Dabei sind Computerprogramme nicht nur als Beschreibung der auszuführenden Funktionen ... Vereinbarung zur Nutzung, ... Dokumentationsinhalte.*

> Software beschreibt die Umsetzung von Algorithmen, Datenstrukturen, Nutzerinterfaces usw. in Code, der auf einem Computer ausgeführt werden kann.

Lebenszyklus einer Software

> Ein Software-Lebenszyklus beschreibt den gesamten Prozess der Herstellung und des Betriebs Implementierung ausgehend von der kundenseitigen Problemstellung über die Realisierung und den Betrieb bis hin zur Ablösung der Software durch einen Nachfolger.

1. Problemstellung
2. Analyse Entwurf
3. Implementierung
4. Test
5. Markteinführung
6. Pflege/Wartung

Welche Querbeziehungen ("Während der Tests wird erkannt, dass die Implementierung Mängel aufweist.") zwischen den einzelnen Stufen sehen Sie?

Welche Definitionen ergeben sich daraus für den Entwicklungsprozess?

> "Unter dem Begriff Softwareentwicklung versteht man die Konzeption und standardisierte Umsetzung von Softwareprojekten und die damit verbundenen Prozesse." [Freund, S. 25]

... oder insbesondere auf große Projekte zielend

> „Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen.“ [Balzert, S. 36]

[Balzert] Helmut Balzert: Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering, 2009

[Freund] Tessen Freund: Software Engineering durch Modellierung wissensintensiver Entwicklungsprozesse, Dissertation, Link google books

Methodische Ziele

Was heißt das, "ingenieurmäßig" oder "standardisiert"?

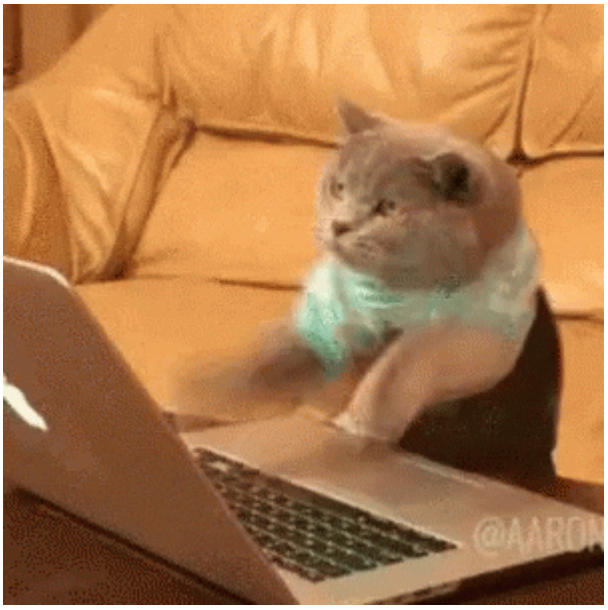
Gemäß ISO 9126 gibt es die sechs folgenden Qualitätsmerkmale für Softwareprodukte:

Die Qualitätskriterien für Software als Produkt nach ISO 9126 [Wiki9126]

[Wiki9126] Nachfolger ISO 25010: Zusätzlich * Kompatibilität * Sicherheit Die Norm kann als eine Art Checkliste verstanden werden. [^Wiki9126]

Von Sae1962 - Eigenes Werk, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=52216179

Und warum der ganze Aufwand?



1940er-1950er: wird der Begriff "Programmierung" verwendet (Software wird meist direkt in Maschinensprache oder Assembler geschrieben)

1960er: massive Probleme bei der Softwareerstellung ("Softwarekrise")

> "Die Hauptursache für die Softwarekrise liegt darin begründet, dass die Maschinen um einige Größenordnungen leistungsfähiger geworden sind! Um es ganz deutlich zu sagen: Solange es keine Maschinen gab, war Programmierung kein existierendes Problem; als wir ein paar schwache Computer hatten, wurde Programmierung zu einem geringen Problem, und nun, da wir gigantische Computer haben, ist die Programmierung ein ebenso gigantisches Problem." – Edsger Dijkstra: The Humble Programmer [Dijkstra]

1968: findet NATO Software Engineering Conference statt, "Software Engineering" wird ersmalig als Konzept diskutiert

[Dijkstra] Edsger Dijkstra: The Humble Programmer (https://www.cs.utexas.edu/~EWD/ewd03xx/EWD340.PDF)

Komplexität von Software

Steigende Komplexität der Softwareprodukte ...

Projekt/Produkt	Lines of Code	Jahr
Unix v 1.0	10.000	1971
Win32/Smile Virus	10.000	2002
Space shuttle	400.000	
Windows 3.1	2.300.000	1992
HD DVD Player (XBox)	4.500.000	2001
Firefox	9.900.000	2010
Android	12.000.000	
F-35 Flugzeug	24.000.000	2013
Facebook	62.000.000	
Autonomes Fahrzeug	100.000.000	

Quelle unter anderem [Weforum] und [McCandless]
Und wann entsteht der Aufwand? Wann muss ein Team Kosten in die Entwicklung investieren?

Projektphase		Relativer Kosteanteil
Spezifikation und Architekturentwurf	Entwicklung	16%
Detailentwurf und Kodierung		8%
Test		16%
Anpassung	Wartung	12%
Erweiterung und Verbesserung		36%
Fehlerbehebung		12%

Zahlwerte aus einem Diagramm in [Lemburg]

Merke: Die Entwicklung kleiner Programme unterscheidet sich von der Entwicklung großer Programme!

Kriterium	Kleine Programme	Große Programme
Zeilenzahl	bis zu ein paar 1000 Zeilen	Millionen von LOC
Einsatz	"Eigengebrauch"	kommerzieller Einsatz von Dritten
Anforderungsanalyse	vage Idee	präzise Spezifikation
Vorgehensmodell	unstrukturiert	strukturierter Entwicklungsprozesse
Test und Validierung	unter Realbedingungen am Endprodukt	Systematische Prüfstrategie
Komplexität	Überschaubare Zahl von Komponenten, Abhängigkeiten usw.	Hohe Komplexität, explizite Organisation in Struktureinheiten und Modulen
Dokumentation	Fehlt in der Regel	zwingend erforlderlich, permanente Pflege
Planung und Organisation	Kaum Planung und Projektorganisation	zwingend erforderlich

Darstellung entsprechen motiviert aus [Lemburg2].

[Lemburg2]	Prof. Dr. Thorsten Lemburg, Einführung in die Softwareentwicklung - Seminar: Softwareentwicklung in der Wissenschaft, Link
[McCandless] David McCandless [https://informationisbeautiful.net/visualizations/million-lines-of-code/)] (https://informationisbeautiful.net/visualizations/million-lines-of-code/) [^Weforum]	Dragan Radovanovic, Kif Leswing, "Google runs on 5000 times more code than the original space shuttle", 2016, Link

Fehler in der Softwareentwicklung

Die Zusammenfassung wurde durch die Ausführungen von [Lemburg] motiviert

1. Management
    - Es wird mit der Codierung sofort angefangen.
    - Eine Festlegung der Anforderungen/Qualitätsmerkmale fehlt.
    - Eine Abnahme der Phasenergebnisse erfolgt nicht.
    - Ein Vorgehensmodell fehlt, bzw. wird nicht verfolgt.
    - Die Schulung für die Software-Ersteller und -Anwender wird vernachlässigt oder als nicht notwendig angesehen.
    - Die Terminvorgaben sind unrealistisch und nicht koordiniert.
    - Begriffe werden nicht definiert.
2. Architektur
    - Die Systemarchitektur ist nicht oder nur sehr umständlich erweiterbar (fehlende Datenkapselung, fehlende Modularität).
    - Es wird ein unnützes Maß an Komplexität in den Entwurf integriert: "*Es könnte doch sein, dass ...*".
3. Entwicklungfluss
    - Die Auswahl der Werkzeuge/Methoden ist unzureichend vorbereitet.
    - Es wird nicht systematisch bzw. unzureichend getestet.
    - Standards und Richtlinien werden nicht beachtet.
4. Dokumentation
    - Schlechte Namensvergabe wie z.B. File-, Klassen-, Methoden- und Variablennamen.
    - Die Dokumentation fehlt bzw. ist veraltet, unzureichend oder nicht adäquat.

Link

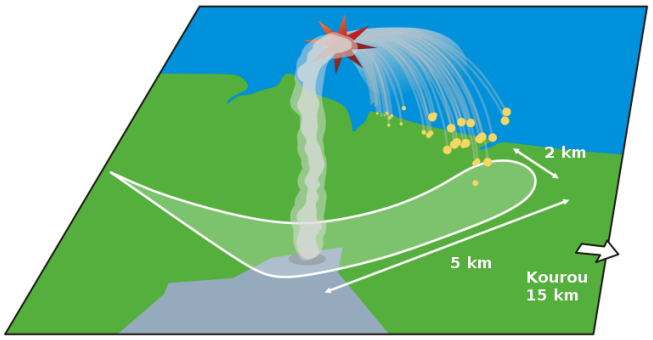
[Lemburg] Prof. Dr. Thorsten Lemburg, Einführung in die Softwareentwicklung,

Konsequenzen von Fehlern im Prozess

> **Ariane Jungfernflug**

V88 war die Startnummer des Erstflugs der europäischen Schwerlast-Trägerrakete Ariane 5 am 4. Juni 1996. Die Rakete trug die Seriennummer 501. Der Flug endete etwa 40 Sekunden nach dem Start, als die Rakete nach einer Ausnahmesituation in der Software der Steuereinheit plötzlich vom Kurs abkam und sich kurz darauf selbst zerstörte. Vier Cluster-Forschungssatelliten zur Untersuchung des Erdmagnetfelds gingen dabei verloren (Schaden 290 Millionen Euro).

[Ariane88]




```
-- Overflow is correctly handled for the vertical component
L_M_BV_32 := TBD.T_ENTIER_16S((1.0 / C_M_LSB_BH) *
                              G_M_INFO_DERIVE(T_ALG.E_BH));
if L_M_BV_32 > 32767 then
 P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;      -- largest 16Bit number (Two's
   complement)
elseif L_M_BV_32 < -32768 then
 P_M_DERIVE(T_ALG.E_BV) := 16#8000#;      -- smallest negative 16Bit number
else
 P_M_DERIVE(T_ALG.E_BV) := UC_16S_EN_16NS(TBD.T_ENTIER_16S(L_M_BV_32));
end if;

-- But not for the horizontal one
P_M_DERIVE(T_ALG.E_BH) := UC_16S_EN_16NS(TBD.T_ENTIER_16S
                              ((1.0 / C_M_LSB_BH) *
                              G_M_INFO_DERIVE(T_ALG.E_BH));
```

Mars Rover

Mars Pathfinder war ein US-amerikanischer Mars-Lander, der 1996 von der NASA eingesetzt wurde. Er brachte 1997 den ersten erfolgreichen Mars-Rover Sojourner auf die Marsoberfläche.

Blick vom Mars Lander auf die Oberfläche des Planeten [NasaMars]

Nach dem Beginn der Aufzeichnung von meteorologischen Daten mit dem Sojourner traten plötzlich scheinbar zufällige System-Zurücksetzungen auf. Das Betriebssystem bootete neu, was mit einem Datenverlust einher ging. Diese Fehler waren aber auch schon auf der Erde aufgetreten ...

Durch Analyse des Logbuches zu diesem Zeitpunkt konnte festgestellt werden, dass es bei der Programmierung von "Sojourner" ein Problem gab. Dabei schlug die sogenannte Prioritäten-Inversion zu, die sich zeigt, wenn mehrere Prozesse ein und die selbe Ressource nutzen.

Weitere Informationen unter What the media coudn't tell you about Mars Pathfinder

> **Aggressiver Gandhi**

Der als friedlich bekannte Inder Mahatma Gandhi ist im Spiele Civilization 5 dafür bekannt, besonders gerne nukleare Waffen zu nutzen. Diese Affinität ist einen Programmierfehler im ersten Teil zuzuordnen, in welchem der Agressionswert bestimmt, wie wahrscheinlich es ist, dass der Herrscher eine atomare Waffe benutzt. Gandhi startet dort mit einem Aggressionswert von 1, jedoch bekommt jede Demokratie bei Spielstart -2 Aggresionspunkte, was zu einem Wert von -1 führt. Binär betrachtet entspricht das folgender 8 Bit Zahl:

1111 1111

Dieser Wert wird intern aber als ein unsigned char, also wie eine 8 Bit vorzeichenlosen Ganzzahl behandelt. Dies führt dazu, dass nicht eine -1 gelesen wird, sondern der Maximalwert dieses Datentyps 255.

[Ariane88] Wikimedia, Autor Phrd, Fragment fallout zone of failed Ariane 501 launch. [Link] (http://www.esa.int/esapub/bulletin/bullet89/images/dalm89f4.gif, https://commons.wikimedia.org/wiki/File:Ariane_501_Fallout_Zone.svg)

[NasaMars] wikimedia, Autor NASA/JPL, Panoramic image from Mars Pathfinder mission, public domain, as NASA is a government institution, Link

Und im Kleinen ...

Das folgende anschauliche Beispiel und die zugehörige Analyse ist durch ein Beispiel der Vorlesung "Software Engineering" von Prof. Dr. Schürr, (TU Darmstadt) motiviert.

> **Achtung:** Das folgende Codebeispiel enthält eine Fülle von Fehlern!

AllesFalsch.c

```c
#include <stdio.h>

FILE *fp;

void main()
{
    fp = fopen("numbers.txt", "r");
    int a[10];
    int num = 0, l = 0;

    while(1){
      if (fscanf(fp, "%d", &num) == 1) {
          a[l] = num;
          l++;
      } else {
          break;
      }
    }
    for(int i=0; i<l; i++)
        printf("%5i ",a[i]);
    printf("\n");

    int aux;
    for(int i=2; i<l; i++){
      for(int j=l; j>i; j--){
        if (a[j-1] > a[j]){
          aux = a[j-1];
          a[j-1] = a[j];
          a[j] = aux;
        }
      }
    }
    for(int i=0; i<l; i++)
        printf("%5i ",a[i]);

    printf("\nAus Maus!\n");
}
```

Aufgabe: Lesen Sie den Code, erklären Sie die Aufgabe, die er löst und identifizieren Sie Fehlerquellen.

Welche Probleme sehen Sie im Hinblick auf die zuvor genannten Qualitätsmerkmale

Aspekt	Bewertung
Funktionalität	?
Zuverlässigkeit	
Benutzbarkeit	
Effizienz	
Wartungsfreundlichkeit	
Übertragbarkeit	

Aspekt	Bewertung
Funktionalität	feste Feldlänge, das Programm stürzt bei mehr als 10 Einträgen ab
Zuverlässigkeit	keine Überprüfung der Existenz der Datei, kein Schließen der Datei
Benutzbarkeit	im Programmcode enthaltene Dateinamen, feste Feldlänge
Effizienz	quadratischer Aufwand der Sortierung
Wartungsfreundlichkeit	fehlende Dokumentation, unverständliche Variablenbezeichner, redundante Codeelemente
Übertragbarkeit	

Herausforderungen

Warum ist Softwareentwicklung so herausfordernd

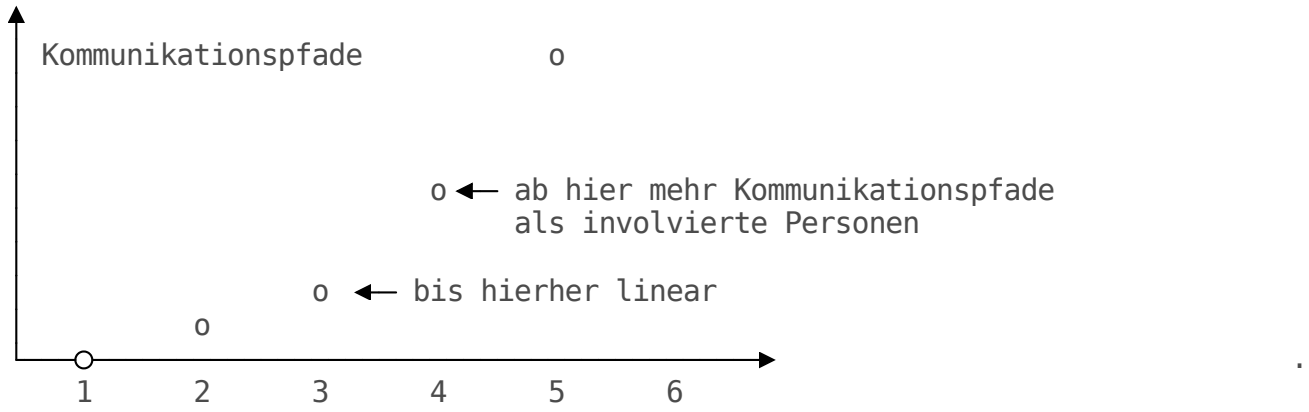
- Die Größe der zu lösenden Probleme. Software ist nicht einfacher, als die Probleme, die sie löst. Je größer und schwieriger die Software, desto aufwendiger und schwieriger ist die Entwicklung.
- Die Tatsache, dass Software ein immaterielles Produkt ist. Die Immaterialität macht das Arbeiten mit Software schwieriger als dasjenige mit materiellen Produkten vergleichbarer Komplexität, da die Risiken auch schwerer zu erkennen sind.
- Sich permanent verändernde Ziele aufgrund der Evolution. Schon das Bestimmen und Erreichen fixierter Ziele bei der Entwicklung ist keine leichte Aufgabe. Sich verändernde Ziele machen das ganze nochmal um eine Größenordnung schwieriger.
- Fehler infolge von Fehleinschätzungen zur Skalierung ("was im Kleinen geht, geht genauso im Großen"). Software-Entwicklung wird daher unbewusst meist als viel einfacher eingeschätzt, als sie tatsächlich ist. Dies führt zu unrealistischen Erwartungen und zu von Beginn an zu tiefen Kosten- und Terminschätzungen.
- Funktionierende Einzelkomponenten stellen noch lange kein funktionierendes Gesamtsystem sicher.

Der Faktor Mensch

In einem Projekt stellt sich die Frage wie viele Personen involviert sind und damit welche Komplexität der Entwicklungsfluss hat. In einem Team von 3 Personen wird muss sichergestellt sein, dass eine Information bei 2 Partnern ankommt. Die maximale Anzahl der Kommunikationspfade ergibt sich zu

$$N = \frac{n \cdot (n-1)}{2}$$

.



.

Grafik motiviert aus [Lemburg2]

Hier ist eine koordinierte Interaktion und Kommunikation notwendig!

Methoden der Aufwandschätzung

- Expertenschätzung - subjektiv, erfahrungsbasiert (Delphi-Methode - Diskussionsrunden, T-Shirt-Sizing, Planning Poker - Schätzung mit verdeckten Karten, anschließend Diskussion)
- Algorithmische Modelle
  - Schätzung über Function-Points: basierend auf Ein- und Ausgaben
  - COCOMO (Constructive Cost Model): Anzahl von Codezeilen
  - UseCase Points (UCP): basierend auf der Anzahl und Komplexität der Anwendungsfälle
- Empirische und vergleichende Methoden (Analogiemethode, Story Points - basierend auf Komplexität, Unsicherheiten und technische Herausforderungen)

Optimale Entwicklungsdauer = 2,5 · (Aufwand in MM)^s mit

s = 0,38 für Stapel-Systeme

s = 0,35 für Dialog-Systeme

s = 0,32 für Echtzeit-Systeme

Brooks-Gesetz:

"Der Einsatz zusätzlicher Arbeitskräfte bei bereits verzögerten Softwareprojekten verzögert sie nur noch weiter." – Frederick P. Brooks: The Mythical Man Month: Essays on Software Engineering

Gründe nach Brooks:

- Einarbeitung neuer Team-Mitglieder kostet Kraft
- Neue Arbeitsaufteilungen sorgen für Unruhe
- Kommunikationskosten steigen im Quadrat mit der Personenzahl

---

[Lemburg2] Prof. Dr. Thorsten Lemburg, Einführung in die Softwareentwicklung - Seminar: Softwareentwicklung in der Wissenschaft, Link

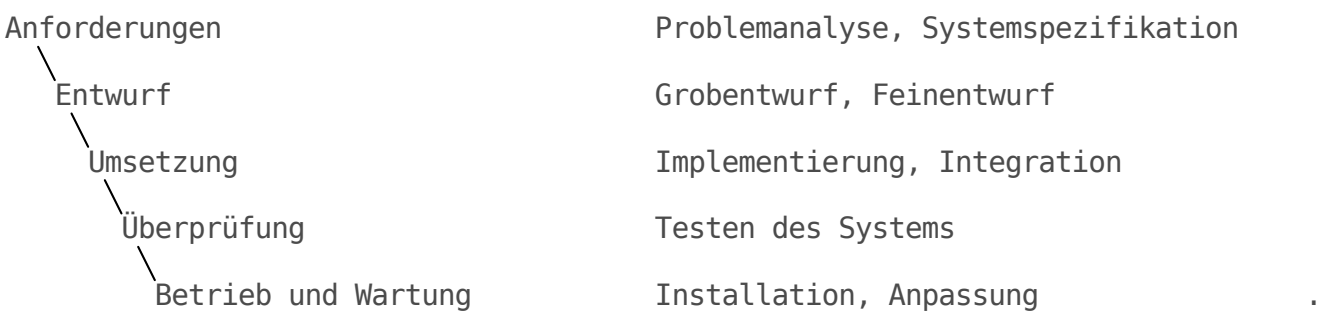
Ansätze zur Strukturierung der Aufgaben

> Ein Vorgehensmodell zur Softwareentwicklung ist ein für die Softwareentwicklung angepasstes Vorgehensmodell bei der professionellen („ingenieursmäßigen“) Anwendungsentwicklung. Es ist ein standardisierter, organisatorischer Rahmen für den idealen Ablauf eines Entwicklungsprojektes und dient dazu, die Softwareentwicklung übersichtlicher zu gestalten und in der Komplexität beherrschbar zu machen. [WikiVorgehen]

Die zwei erstgenannten Vorgehensweisen setzen auf eine Planbarkeit des Projektes, während die agilen Methoden die Flexibilität in den Vordergrund stellen.

[WikiVorgehen] Wikipedia

Wasserfallmodell



Eigenschaften des Wasserfallmodells:

- Aktivitäten sind in der vorgegebenen Reihenfolge und in der vollen Breite vollständig durchzuführen.
- Am Ende jeder Aktivität steht ein fertiggestelltes Dokument, d.h. das Wasserfallmodell ist ein „dokumentgetriebenes“ Modell.
- Der Entwicklungsablauf ist sequentiell und als Top-down-Verfahren realisiert.
- Es ist einfach, verständlich und benötigt nur wenig Managementaufwand.

Vorteile:

- klare Abgrenzung der Phasen – einfache Möglichkeiten der Planung und Kontrolle
- bei stabilen Anforderungen und klarer Abschätzung von Kosten und Umfang ein sehr effektives Modell

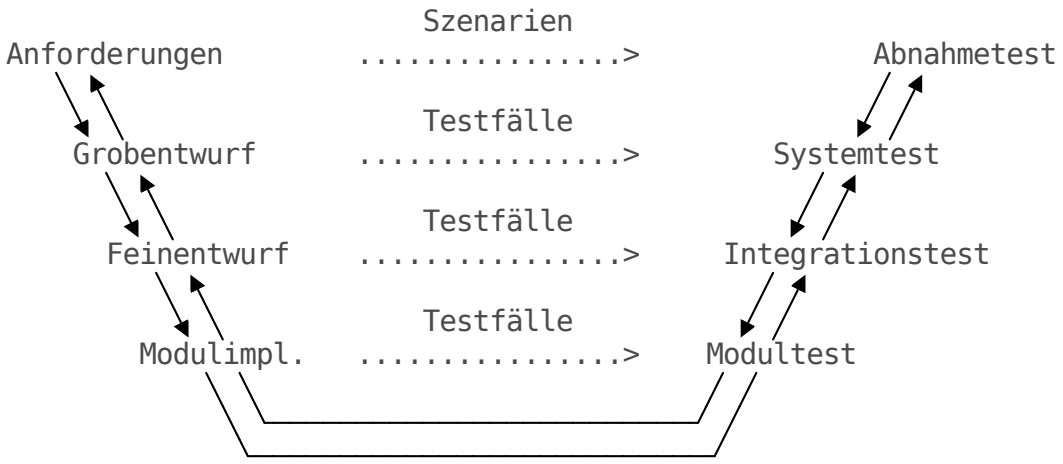
Nachteile:

- Das Modell ist nur bei einfachen Projekten anwendbar – Unflexibel gegenüber Änderungen und im Vorgehen
- Frühes festschreiben der Anforderungen ist sehr problematisch und kann zu teuren Änderungen führen
- Fehler werden eventuell erst sehr spät erkannt und müssen mit erheblichen Aufwand entfernt werden

Im erweiterten Wasserfallmodell fällt die strikte Vorgabe eine Phase nach der anderen zu bearbeiten weg, die Rückkehr in eine vorhergehende Phase möglich, um z.B. Fehler zu beheben.

V-Modell

Zusätzlich zu den Entwicklungsphasen definiert das V-Modell das Vorgehen zur Qualitätssicherung (Testen), indem den einzelnen Entwicklungsphasen die Testphasen gegenübergestellt werden.



Tätigkeitsbereiche des V-Modell: Softwareerstellung, Qualitätssicherung (Reviews, Tests, Verifikationen bzw. Validierungen), Konfigurationsmanagement (Verwaltung und Kontrolle von Versionen, Änderungen und Abhängigkeiten), Projektmanagement (Zeitmanagement, Ressourcenplanung, Risikomanagement und Kommunikation)

Vorteile:

- Integrierte und detaillierte Darstellung von den Tätigkeitsbereichen
- Generisches Modell mit definierten Möglichkeiten zur Anpassung an projektspezifische Anforderungen
- Gut geeignet für große Projekte

Nachteile:

- Für kleine und mittlere Softwareentwicklungen führt das V-Modell zu einem unnötigen Overhead
- Die im V-Modell definierten Rollen (bis zu 25: Projektleiter, Auftragsgeber, Qualitätsmanager, Systemarchitekt, Softwareentwickler, Tester / Testmanager, ...) sind für gängige Softwareentwicklungen nicht realistisch
- explizite Werkzeuge notwendig

Agile Softwareentwicklung

Wesentliche Gründe für agile Herangehensweisen sind, dass sich die Ziele und das Umfeld (beteiligte Personen, Marktanforderungen, technisches Umfeld/Schnittstellen) im Laufe des Projektes ändern. Die agilen Methoden eignen sich daher besonders gut, um auf geänderte Anforderungen zu reagieren, da die Entwicklungszyklen in der Regel kurz angelegt sind. Die Anforderungen werden häufig nur knapp beschrieben und erst kurz vor Beginn von Umsetzung und Test ausformuliert. Durch die kurzen Zeiträume sind (nachträgliche) Änderungen der Anforderungen relativ leicht möglich.

	Klassische Methoden	**Agile Methoden**
Anforderungen am Projektbeginn	klar definiert	unscharf
Änderungsbereitschaft	gering	explizit
Anforderungsbeschreibung	technische Sicht / Perspektive des Unternehmens	Kundensicht (Anwendungsfälle)
Entscheidungsfindung	Gremien	Team
Planung	durch Projektleiter	im Team
Kundeninteraktion	Kunde oft unbekannt	Kunde als aktiver Teil

Agile Leitsätze

Vier Leitsätze wurden im Februar 2001 als Agiles Manifest formuliert:

"Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen.

Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

- Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge
- Funktionierende Software ist wichtiger als umfassende Dokumentationen
- Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlungen
- Reagieren auf Veränderung ist wichtiger als das Befolgen eines Plans

Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein." – Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern,

Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland und Dave Thomas [Manifesto]

Agile Methoden sind z.B.:

- Scrum: Strukturierter, iterativer Ansatz für Teams
  - Sprints (Zeitboxen von 1–4 Wochen) mit klaren Zielen
  - Daily Stand-ups zur Synchronisation
  - Sprint Reviews & Retrospektiven zur Verbesserung
- Extreme Programming (XP): legt Wert auf technische Exzellenz, Code-Qualität und Best Practices, stark entwicklerorientiertes Model
  - Test-Driven Development (TDD) - erst Test dann Code
  - Pair Programming (Arbeiten in Zweierteams)
  - Continuous Integration (automatisierte Builds und Tests nach jeder Änderung) und schnelle Releases
- Kanban
- Crystal

[Manifesto] Tessen Freund: Software Engineering durch Modellierung wissensintensiver Entwicklungsprozesse, Dissertation, Manifesto

Ok, wir brauchen Unterstützung

> *CASE (Computer-Aided Software Engineering) is the use of computer-based support in the software development process*

Was können die CASE-Tools?

- Aufgaben wie Diagrammerstellung, Codegenerierung und Dokumenterstellung automatisieren.
- Durch Reduzierung des Entwicklungsaufwandes die Entwicklungszeit verkürzen.
- Softwarequalität verbessern.
- Verwaltung von Softwareprojekten organisieren.
- Zusammenarbeit zwischen Teammitgliedern verbessern.

Klassifikation nach dem Einsatzzweck

- Anforderungsanalyse
  - Spezifikation
  - Modellierung
- Code-Erstellung (Editoren)
  - Editor, IDE
  - Dokumentation
- Ausführung und Testen
  - Compiler, Interpreter
  - IDE, Build-System
  - Debugger
- Koordination Entwicklungsprozess
  - Projektverwaltung
  - Code-Base Management und Versionierung
  - Deployment
  - Support

Grad der Integration verschiedener Entwicklungswerkzeuge

- Tools - spezialisierte Werkzeuge für einzelne Aktivitäten im Software Life-cycle
- Workbenches - Sammlungen mehrerer Tools für bestimmte Aufgaben (z. B. Modellierungs-, Test-Frameworks)
- Integrated Development Environments (IDEs) - vollständig integrierte Entwicklungsumgebungen, die den gesamten Software-Lifecycle unterstützen (z. B. VS Code, Eclipse, IntelliJ)

> Texteditor vs. Integrated Development Environment (IDE) ... worfür soll ich mich entscheiden?

- Analyse des Workflows und der Formen der Zusammenarbeit (agil oder klassisch?)
- Analyse der verwendeten Spezifikations und Modellierungstechniken (wird Modelierungstool benötigt?), Programmiersprachen (Debugging? Code-Vervollständigung?), etc.
- Analyse der Komplexität des Vorhabens, der erforderlichen Unterstützung mit Versionskontrolle, Debugger, Build-Tools
- Analyse der Rahmenbedingungen (Betriebssysteme, Kosten)

Aufgaben

- [ ] Betrachten Sie die Darstellung unter Webseite Programmwechsel und versuchen Sie die überspitzten Missverständnisse der einzelnen Protagonisten im Kontext eines Softwareprojektes zu stellen.
- [ ] Korrigieren Sie das `allesFalsch.c` Beispiel, verbessern Sie die Lesbarkeit des Codes.

Quizze

Wo entsteht der größte Zeit- bzw. Kostenaufwand bei kommerzieller Software?

- [ ] Entwurf
- [ ] Programmierung
- [ ] Service/Wartung

Wann ist eine Software-Dokumentation wichtig?

- [ ] immer, man sollte auch bei kleinsten Projekten eine möglichst detaillierte Dokumentation schreiben
- [ ] bei (mittel-)großen Projekten
- [ ] bei kleinen Projekten mit komplexem Code
- [ ] die Dokumentation ist generell nicht wichtig, erfahrene Programmierer finden sich selber zurecht