Vorlesung "Softwaretechnik"

Die Welt der Softwaretechnik

Lutz Prechelt
Freie Universität Berlin, Institut für Informatik

- Routine und Innovation
  - "Normales Vorgehen", "Radikales Vorgehen"
- Taxonomie: Die Welt der Softwaretechnik
  - Probleme
    - Produkt: technische P.
    - Prozess: psycho-soziale P.
  - Lösungsansätze
    - "harte" technische
    - "weiche" methodische

Lernziele

- Verstehen, warum Ingenieurwesen auf erfahrungsgeleitetes Vorgehen angewiesen ist
  - *"Normales Vorgehen"*: Wünschenswert!
- Verstehen, warum man in seinem Vorgehen immer irgendwo von vorherigen Erfahrungen abweicht
  - *"Radikales Vorgehen"*: Heikel!
- Ansätze für normales Vorgehen verstehen:
  - Was sind die typischen **Probleme** bei der SW-Entwicklung?
  - Was sind die dazu passenden **Lösungsansätze** der SWT?
- Eine Unterteilung der Softwaretechnik verstehen:
  - "Die Welt der Softwaretechnik"

Fallbeispiel: Die Tacoma Narrows Bridge (TNB)

- 1938–1940: Brückenbau über den Puget Sound im Staat Washington
  - und zwar als Hängebrücke
  - wegen großer Spannweite
    - nicht verwechseln mit Gesamtlänge!

Balkenbrücke (bis ~200m)
Girder bridge

Bogenbr. (bis ~150m)
Arch bridge

Fachwerkbr. (bis ~500m)
Truss bridge

Schrägseilbr. (bis ~500m)
Cable-stayed bridge

Hängebr. (bis ~1000m)
Suspension bridge

(http://www.matsuo-bridge.co.jp/english/bridges/index.shtm)

TNB: Chefingenieur

- Chefingenieur war Clark Eldridge
- Chefberater für die Konstruktion war ein Experte für Hängebrücken, Leon Moisseiff
  - Er hatte zuvor schon die Manhattan Bridge, Bay Bridge, George Washington Bridge und die Golden Gate Bridge maßgeblich mitgebaut
  - und war Spezialist für die Rechenmodelle

Golden Gate Bridge
1970m Länge
1280m Spannweite

Bay Bridge (west)
2820m Länge
700m Spannweite

Tacoma Narrows B.
1520m Länge
850m Spannweite

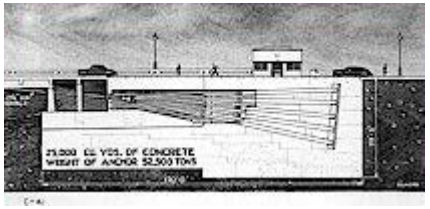
TNB: Konstruktionsparameter

Es gibt eine Vielzahl wichtiger Konstruktionsparameter für eine Hängebrücke:

- z.B. für die Einzelteile
  - Pylonverankerung, Pylon, Tragseilverankerung, Tragseil, Hängeseil, Fahrbahnunterbau, Fahrbahn
  - (jedes dieser Teile besteht natürlich wiederum aus einer Vielzahl von Teilen)
- jeweils diverse immer relevante Parameter
  - Material, Bauform, Verarbeitungsweise, Dimensionen
- und zusätzlich weitere spezielle (je nach Teil)
- Ferner globale Parameter

TNB: Konservative Konstruktionsparameter

Die Tacoma Narrows Bridge hatte überwiegend ganz übliche Parameter für die Einzelteile:



- Übliche Pylonen
  - bezügl. Bauart, Größen, Materialien etc.
- übliche Trag- und Hängeseile
  - bezügl. Bauart, Größen, Materialien etc.
- eine übliche Verankerung
  - bezügl. Bauart, Größen, Materialien etc.
- eine übliche Fahrbahn
  - bezügl. Bauart, Größen (2 Fahrspuren, 2 Gehwege), Materialien
- einen weitgehend üblichen Unterbau
  - bezügl. Größen, Materialien

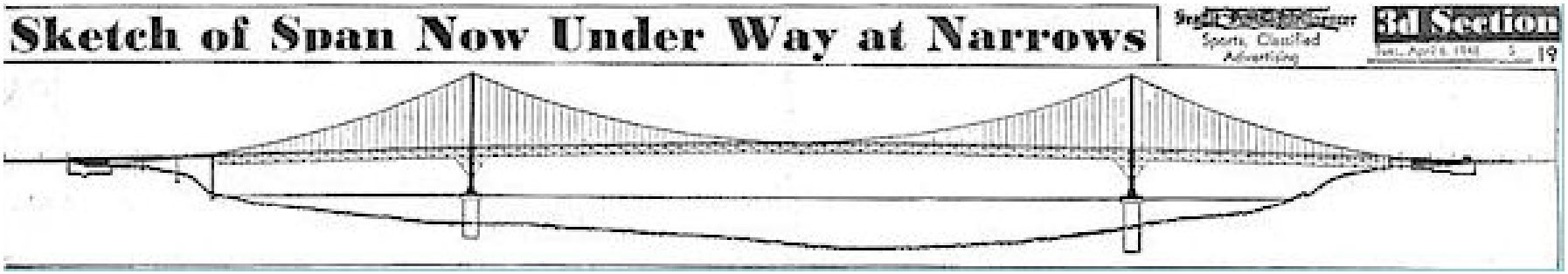
TNB: Ungewöhnliche Konstruktionsparameter

- Spannweite
  - Drittgrößte je gebaute (bis dahin)
  - Aber Moisseiff hatte die einzigen beiden größeren (Golden Gate, George Washington) mitgebaut
- Konstruktion des Fahrbahn-Unterbaus ("Tragwerk"):
  - Geschlossene Seitenwand statt Fachwerk
- Schlankheit
  - Tragwerk nur 12m breit und 2,40m hoch
  - Das Verhältnis Tragwerkbreite:Spannweite war 1:72 (Golden Gate Bridge: 1:47)
  - Das Verhältnis Tragwerkhöhe:Spannweite war 1:350 (üblich war eher bis zu 1:90)

TNB: Entscheidung über Parameter

- Der Entwurf wurde für diese ungewöhnlichen Parameter von anderen Brückeningenieuren vor dem Bau kritisiert
  - Sie war ungewöhnlich leicht und flexibel für ihre Spannweite
    - geringe Breite, geringe Höhe des Tragwerks (also wenig Steifheit)
  - und Hängebrücken sind prinzipbedingt seitenwindempfindlich

- Aber Moisseiff bestand darauf, sie würde funktionieren
  - Er hatte ein Rechenmodell, nach dem sie Winden bis über 140 km/h standhalten sollte
  - Die Geldgeber verlangten daraufhin die leichte Konstruktion


Sketch of Span Now Under Way at Narrows

Sports, Classified Advertising

3d Section

TNB: Bau und Eröffnung, Verhalten

- Bau ging reibungslos vonstatten
- Eröffnung erfolgte im Juli 1940

- Schon während des Baus zeigte die Brücke starke Bewegungen bei Wind
  - Spitzname "Galloping Gertie"
- Nicht nur Querschwingungen, sondern achterbahnartige Längsschwingungen
  - Konstruktion war aber offenbar stark und flexibel genug, keine Schäden
- Am 7. November 1940 schaukelten sich die Schwingungen soweit auf, dass die Brücke zerbrach
  - Es folgt ein Originalfilm der Katastrophe
  - Kein Zeitraffer o.ä.!

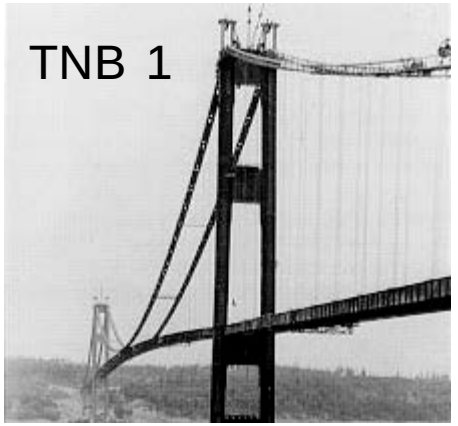
TNB: Der Kollaps

TNB: Ursache für den Kollaps

- Keine Ursache: Die Windstärke
  - Sie war mit ca. 75 km/h noch weit im zulässigen Bereich
- Unfallhergang (Mechanismen):
  - Durch die Schwankungen rissen einige der Hängeseile, was zur Überlastung der benachbarten Hängeseile führte u.s.w.
- Unfallursache (Bedingungen):
  - Die Längsschwingungen, für die die Brücke nicht konstruiert war
- Unfallherkunft (Urgründe):
  - Herkunft: Das Rechenmodell von Moisseiff berücksichtigte nur Querschwingungen
  - Urgrund: Längsschwingungen waren vor TNB nie relevant gewesen, weil vergleichbar lange Brücken stets viel schwerer und steifer gebaut worden waren
  - **Urgrund: Moisseiffs Konstruktion lag zu weit außerhalb des Bereichs, über den es Erfahrungen gab**

TNB Version 2

- 1948-1950 wurde eine neue Tacoma Narrows B. gebaut
  - Gleiche Spannweite
    - Fundamente wiederverwendet
  - Die Tragwerkbreite wurde um 50% erhöht
    - von 12m auf 18m
  - Die Tragwerkhöhe wurde um 300% erhöht
    - von 2,4m auf 10m
  - Das Tragwerk ist wieder ein offenes Fachwerk



- Die neue Brücke liegt damit für alle Parameter wieder im normalen Bereich:
  - Verhältnis Tragwerkhöhe:Spannweite sinkt auf übliche 1:85
  - Verhältnis Tragwerkbreite:Spannweite sinkt auf übliche 1:48 (wie bei Golden Gate B.)

TNB: Quellen

- http://www.lib.washington.edu/specialcollections/collections/exhibits/tnb
  - Hauptquelle, ausführliche Dokumentation mit vielen Bildern
  - ziemlich interessant!
- Vergleich der Parameter von TNB 1 und TNB 2
  - http://www.nwrain.net/~newtsuit/recoveries/narrows/comp.htm
- Englische Wikipedia:
  - Tacoma Narrows Bridge
  - Golden Gate Bridge
  - Bay Bridge
- Die längsten Brücken verschiedener Typen:
  - http://www.bernd-nebel.de/bruecken/1_einfuehrung/rekorde/rekorde.html

TNB: Zusammenfassung

- Die TNB war eine Konstruktion, die man zu beherrschen glaubte
  - Fast alle Eigenschaften waren wohlverstanden und wurden richtig gemacht
    - Materialien/Verarbeitung/Dimensionierung für Pylone, Tragwerk, Fahrbahn, Tragseil, Halteseile, Verankerung: alles bestens
  - Der weltbeste Experte hatte die Brücke konzipiert
- Sie entfaltete dann jedoch eine Eigenschaft (Längs-schwingungen), mit der niemand gerechnet hatte
  - und die nicht von einem bestimmten Teil herrührte, sondern von deren Zusammenwirken ("**emergente Eigenschaft**")
- und die die Konstruktion komplett zunichte machte
- Ursache: Man war in einer einzigen Hinsicht (Leichtheit) weit von dem abgewichen, was man aus vergleichbaren Projekten kannte

Bitte hinter die Ohren schreiben!

Erfahrung
und emergente Eigenschaften

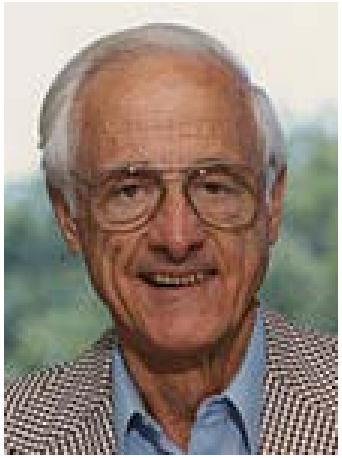
- Die emergenten Eigenschaften komplexer dynamischer Systeme lassen sich durch keine Theorie komplett vorhersagen
  - Man versteht und beherrscht sie nur allmählich aufgrund von Erfahrung
  - https://de.wikipedia.org/wiki/Komplexes_System
- Etwas Neues ist also nur verstehbar/beherrschbar, wenn es Bekanntem genügend ähnelt
- Deshalb ist erfolgreiches Ingenieurwesen davon abhängig,
  - immer möglichst viel so zu machen "wie üblich"
  - und nur möglichst wenig "neu" oder "ganz anders"
  - denn ein einzelnes Projekt soll ja zuverlässig und auf Anhieb klappen; nicht "nur allmählich aufgrund von Erfahrung" (s. oben)

Normales Vorgehen und radikales Vorgehen

Freie Universität Berlin

- Etwas so zu machen "wie üblich" nennen wir **"normales Vorgehen"**
  - N1 Wir wissen, wie das geht. Wir tun es routinemäßig.
  - N2 Wir verstehen, worauf es dabei ankommt (Erfolgsfaktoren)
  - N3 Wir kennen die typischen Probleme dabei (Risikofaktoren)
  - N4 Wir verstehen, wofür das Vorgehen geeignet ist und wofür vielleicht nicht (Anwendbarkeitsbereich)
  - N5 Wir dürfen zuversichtlich sein, einen Erfolg zu erzielen
- Etwas "ganz anders" zu machen oder etwas "ganz Neues" zu tun nennen wir **"radikales Vorgehen"**
  - R1 Wir wissen nur *grob* (allgemein, nicht konkret), wie das geht
  - R2 Wir verstehen nur *teilweise*, worauf es dabei ankommt
  - R3 Wir wissen noch *nicht*, welche Probleme uns bevorstehen
  - R4 Wir verstehen höchstens *ansatzweise* Eignung und Grenzen
  - R5 Wir können allenfalls *hoffen*, einen Erfolg zu erzielen

Herkunft der Begriffe



- Die Unterscheidung wurde geprägt von Walter Vincenti, Professor für Luftfahrtwesen
  - Er nennt es "normal design", weil in aller Welt (außer bei den Softwareleuten) das Wort Design die Anforderungsdefinition mit einschließt

- Hier seine Formulierungen:
  - "[The engineer practicing normal design] knows at the outset how the device in question works, what are its customary features, and that, if properly designed along such lines, it has a good likelihood of accomplishing the desired task."
  - "[In radical design] how the device should be arranged or even how it works is largely unknown. The designer has never seen such a device before and has no presumption of success. The problem is to design something that will function well enough to warrant further development."
    - Walter Vincenti: "What engineers know and how they know it", John Hopkins University Press, 1993

Anmerkung 1: Aspekte (nicht das Ganze) sind normal/radikal

Freie Universität Berlin

- Jeder Aspekt eines Projekts ist entweder normal oder (mehr oder weniger) radikal
  - Jedes Einzelteil des Resultats (Produkt)
  - Jeder Arbeitschritt auf dem Weg dahin (Prozess)

- Kaum ein Ingenieurprojekt kommt ohne Anteile radikalen Vorgehens aus
  - fast immer muss irgendwas anders gemacht werden als gewohnt

- aber ein insgesamt "normales" Projekt hat nur wenige und unbedeutende radikale Anteile/Aspekte
  - oder erwirbt die nötigen Erfahrungen unterwegs

Anmerkung 2: Lernziele in einem Ingenieurfach

Freie Universität Berlin

1. Grundlagenwissen erwerben, um nötiges radikales Vorgehen besser bewältigen zu können
   - z.B. Informatikstudium: gesamtes Grundstudium
   - Allgemeines Wissen: Breit

2. Erfahrungswissen und Urteilskraft erwerben, um möglichst oft normal vorgehen zu können
   - …und radikalem Vorgehen wo immer möglich aus dem Weg gehen zu können
   - Bereichsspezifisches Wissen: Eng!

- Kurz gefasst:
  - **Radikales Vorgehen ist der (einzige) Weg für Anfänger**
  - **Normales Vorgehen ist die Domäne der Profis**

Anmerkung 3: Das Spannungsfeld

- Eigentlich ist also überall normales Vorgehen wünschenswert
- Aber das benötigt viel Erfahrung im jeweiligen Bereich
- Immer normal vorzugehen schließt also hochinnovative Projekte aus
  - denn bei einer Innovation hat man per Definition keine Erfahrung

- Deshalb gibt es ein Spannungsfeld zwischen
  - risikoarmen Projekten (wenig radikales Vorgehen nötig)
    - die können aber nicht sehr innovativ sein
  - innovativen Projekten (viel radikales Vorgehen nötig)
    - die sind aber stets riskant.
    - Merke: Selbst bei hochinnovativen Funktionen nach außen kann eine Software intern technisch oft nichtradikal sein. Sehr wünschenswert!

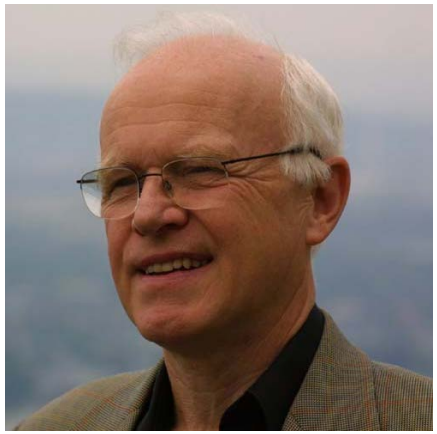
Anmerkung 4: Der Umkehrschluss gilt nicht

- Zwar benötigt ein innovatives Vorhaben einen hohen Anteil radikalen Vorgehens
- aber deshalb heißt viel radikales Vorgehen noch längst nicht, dass man etwas Innovatives tut

- Softwareingenieure neigen bislang dazu, an Stellen radikal vorzugehen, wo das weder sinnvoll noch nötig ist:

Anmerkung 4 (2): "Wo bleibt da die Herausforderung?"

- "Das Wort 'Herausforderung' hat für Software-Entwickler eine geradezu religiöse Bedeutung.
  Herausforderungen sind es, die unserer Arbeit Bedeutung geben und den Unterschied zwischen Eintönigkeit und dem Aufbruch zu neuen Ufern ausmachen.
  Wir verwenden fast ebensoviel Kreativität darauf, nach Herausforderungen zu suchen, wie darauf, ihnen zu begegnen.
  Deshalb neigen Entwickler dazu, selbst in der prosaischsten Aufgabe eine Notwendigkeit für Eleganz oder Prägnanz oder Schlankheit oder Geschwindigkeit oder Flexibilität zu entdecken, die über die spezifizierte Anforderung hinausgeht."



Tom DeMarco in
"Warum ist Software so teuer?", S.83
(auch in IEEE Software Nov. 1990)

Herausforderungen

- Bitte behalten Sie stets die Zweckmäßigkeit im Auge!

Quellen radikalen Vorgehens

- Ziele, unverstandene Anforderungen
  - Wir verstehen nicht gut genug, WAS für ein System wir überhaupt bauen müssen, um den Zweck zu erreichen
  - Was also das geplante Verhalten sein soll
    - Beispiel folgt
- Konstruktion, unverstandener Entwurf
  - Wir verstehen nicht gut genug, WIE das System beschaffen sein muss, damit es sich wie geplant verhält
    - wegen emergenter Eigenschaften
      (unerwartete Schwierigkeiten; Beispiel: Tacoma Narrows Bridge)
    - oder weil es einfach sehr viele Teile braucht
      (klar sichtbare Schwierigkeiten; Beispiel: Jedes große SW-Projekt)
- Prozess, unverstandene Methodik
  - Wir verstehen nicht gut genug die VERFAHREN, mit denen man zu korrekten Anforderungen und tragfähigem Entwurf gelangt
    - Beispiel: das alltägliche Drama in vielen SW-Organisationen

Beispiel
für unverstandene Anforderungen

Freie Universität Berlin

- Ziel: die Radbremsen und die Schubumkehr eines Verkehrsflugzeuges sollen nur nach der Landung ansprechen
  - Vor der Landung sollen sie gesperrt sein
  - Das arrangiert man heute natürlich per Software

- Daraus abgeleitete Anforderungen (Airbus A320):
  - Bremsenfreigabe = beide Haupträder drehen mit mindestens 80% der Bodengeschwindigkeit
    - gemessen durch Drehgeschwindigkeits-Sensor an jedem Rad
  - Schubumkehrfreigabe = beide Haupträder auf dem Boden oder ein Rad auf dem Boden und dreht mit mind. 72 Knoten
    - Bodenkontakt gemessen durch Gewichtsschalter am Fahrwerk
  - (Vereinfacht. In Wirklichkeit noch um einiges komplizierter)

http://www.rvs.uni-bielefeld.de/publications/Incidents/DOCS/ComAndRep/Warsaw/leyman/analysis-leyman.html

Anforderungen A320 Schubumkehr

- Ein Landeanflug:
  - Tower: "13 Knoten Wind von schräg vorn" (also mit Seitenwind)
  - Pilot wählt deshalb genau gemäß Lehrbuch manuelle Landung und Anfluggeschwindigkeit 145 Knoten(Luft) statt normal 130
  - Tatsächlich sind aktuell 15–25 Knoten Rückenwind mit Böen
    - Landegeschwindigkeit steigt von 130 auf 155 Knoten
  - Landung erfolgt zunächst auf nur einem Hauptrad
    - genau gemäß Lehrbuch für Seitenwindlandung
  - Landebahn ist 3mm hoch mit Wasser bedeckt
    → Aquaplaning → Das Rad beschleunigt nur langsam
  - Hohe Geschwindigkeit → langsames Landen
    9 Sekunden Verzögerung bis Freigabe Bremsen/Schubumkehr
    → Flugzeug schießt über Landebahnende hinaus
  - 2 Tote (Warschau, 14. Sept. 1993)

Anforderungen A320 Schubumkehr (2)

- Das computerisierte Steuersystem der A320 war korrekt implementiert und hat genau wie spezifiziert gearbeitet
  - Aber es war nicht gut genug vorhergesehen worden, wie es mit den Umweltbedingungen interagieren könnte

- Grund: die Anforderungen an die Schubumkehrsperre waren ein radikales Vorgehen:
  - R2 Wir verstehen nur *teilweise*, worauf es dabei ankommt
  - R3 Wir wissen noch *nicht*, welche Probleme uns bevorstehen

Wichtigster Ansatz für normales Vorgehen: Wiederverwendung

Freie Universität Berlin

Wiederverwendung

- von bewährten Komponenten
- von bewährten Entwurfsüberlegungen/Entwürfen
- von bewährten Anforderungen
- von bewährten Prozesselementen
- von bewährten Werkzeugen

(Die Übergänge sind fließend)

- Achtung: Wiederverwendung hat zwei Hauptziele
  - Senkung der Kosten
  - Senkung des Risikos
- Aus Sicht des normalen Vorgehens ist nur die Risikosenkung von Interesse
  - Die Kostensenkung gibt es quasi gratis dazu

Anmerkung 5:
Nicht radikal, aber auch ungünstig

Freie Universität Berlin

- Auch normales Vorgehen kann ungünstig sein:
  nämlich unangemessen
- Beispiel:
  - Ich entwerfe und implementiere eine komplexe Komponente,
    die ich sehr gut verstehe, mit normalem Vorgehen,
    - z.B. bekanntem Entwurf und wiederverwendeten Anforderungen
      - z.B. ein Relationales Datenbank-Managementsystem (RDBMS)
  - obwohl mir eine gut geeignete wiederverwendbare Realisierung
    zur Verfügung steht
      - z.B. PostgreSQL, MariaDB, SQlite, HSQLDB, …

- Solches Verhalten ist ineffizient (also teuer)
  und deshalb nicht ingenieurgerecht
  - ähnlich wie ein vermeidbares radikales Vorgehen

Fazit: Routine versus Innovation

- Normales Vorgehen bedeutet, etwas zu tun, das man gut kennt, versteht und beherrscht (Routine)
  - Anwendungsweise, Erfolgsfaktoren, Risikofaktoren, Anwendungsbereich
- Radikales Vorgehen ist alles Übrige

- Wann immer möglich sollte man normales Vorgehen praktizieren
  - oder wenigstens anstreben
- Radikales Vorgehen ist dann akzeptabel, wenn es sich nicht umgehen lässt:
  - z.B. wegen Mangels an Personal mit dem passenden Wissen
    - aber besser wäre: Jemand passendes anheuern
  - z.B. wegen echter Innovation
    - d.h. überhaupt niemand könnte es auf normale Weise tun

Meine Bitte an Sie

1. Bitte bemühen Sie sich darum, das verfügbare Erfahrungswissen zu erlernen
   - (nicht nur technisches Grundlagen- und Spezialwissen)
   - um häufiger normal vorgehen zu können.
   - Bücher und das Web sind enorm hilfreich
     - …wenn man Sie genügend benutzt

2. Sobald Sie im Beruf sind, spezialisieren Sie sich passend auf eine Domäne, um die Quote normalen Vorgehens so hoch wie nur möglich zu treiben
   - und lernen Sie kontinuierlich dazu (neue Anforderungen und neue Technologie), damit sie hoch bleibt.
   - Bücher und das Web sind dazu enorm hilfreich
     - …wenn man Sie genügend benutzt

Fragen an die Softwaretechnik

Aus dieser Sicht sind also zwei Fragen an die Softwaretechnik(vorlesung) zu richten:

1. Welches Wissen für normales Vorgehen ist verfügbar?
   - Wir können das in der Vorlesung nur grob anreißen
   - Hauptquellen sind **Muster** verschiedenster Art
2. Welche Ideen und Methoden helfen dabei, ein (teilweise) nicht-normales Vorgehen möglichst oft zum Erfolg zu führen?
   - Das bildet den Grundstock dieser Vorlesung

- Der Übergang zwischen normalem und radikalem Vorgehen ist fließend!

Wie geht es jetzt weiter?

- Heute: Taxonomie (Einordnung)
  1. der grundlegenden Probleme, die die SWT zu lösen hat
  2. der zugehörigen Lösungsansätze
- Rest der Vorlesung:
  - Vermittlung von Grundwissen über die Lösungsansätze
  - unter Verweis auf die Probleme

- Bitte beachten Sie:
  - Alle Lösungsansätze sind wesentlich genauer ausgearbeitet, als wir hier besprechen können
    - Lernen Sie weiter, wo immer sie etwas davon nutzen
  - Insbesondere gibt es viel mehr wiederverwendbares Wissen
    - Muster, Vorgehensbeschreibungen, Checklisten, Werkzeuge u.a.
  - Fast alle Ansätze müssen je nach Entwicklungssituation unterschiedlich angewendet werden
    - siehe Einführungsvorlesung

Taxonomie
"Die Welt der Softwaretechnik"

Freie Universität Berlin

Welt der Problemstellungen:

- Produkt (Komplexitätsprob.)
  - Anforderungen (Problemraum)
  - Entwurf (Lösungsraum)
- Prozess (psycho-soziale P.)
  - Kognitive Beschränkungen
  - Mängel der Urteilskraft
  - Kommunikation, Koordination
  - Gruppendynamik
  - Verborgene Ziele
  - Fehler

Welt der Lösungsansätze:

- Technische Ansätze ("hart")
  - d.h. wir wissen genau, was wir damit meinen (leider nur sehr allgemein)
  - Abstraktion
  - Wiederverwendung
  - Automatisierung
- Methodische Ansätze ("weich")
  - d.h. wir wissen oft nur vage, was wir damit meinen
  - Anforderungsermittlung
  - Entwurf
  - Qualitätssicherung
  - Projektmanagement

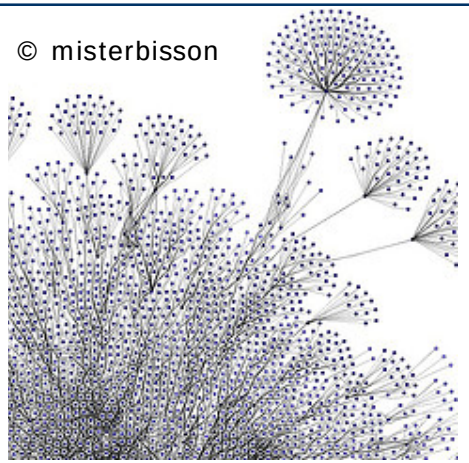
Welt der Problemstellungen

"Mit welchen Phänomenen muss die SWT fertig werden?"

- Enthält all das, was Programmieren-im-Großen unterscheidet vom Programmieren-im-Kleinen

- Alle diese Probleme sind essenziell
  - d.h. sie lassen sich nicht beseitigen,
  - sondern nur abmildern.
  - Berühmter Aufsatz dazu:
    Fred(erick) Brooks: *"No Silver Bullet: Essence and Accidents of Software Engineering"*,
    IEEE Computer 20(4), April 1987,
    www.computer.org
  - Seine Vorschläge: Kaufen statt bauen; Prototyping; Inkrementelle Entwicklung; Herausragende Entwerfer/innen

Produktprobleme (Komplexitätsprobleme)



- Komplexität ("Verflochtenheit") bedeutet
  - ein System besteht aus vielen Einzelteilen und
  - es hat emergente Eigenschaften
- Emergente Eigenschaften:
  - Phänomene, die man nicht verstehen kann, indem man die Einzelteile betrachtet
  - sondern die erst aus deren Zusammenwirken entstehen
- "Komplex" heißt also praktisch so viel wie
  - "schwierig aufgrund vielfältiger Zusammenhänge"
- Komplexitätsprobleme bei SW-Entwicklung stammen meist aus der Komplexität des Produkts und liegen
  - im Problemraum (Anforderungen; Was wollen wir bauen?)
  - oder im Lösungsraum (Entwurf; Wie sollen wir es strukturieren?).

  Meistens in beiden.
  - Prozesskomplexität ist vor allem eine Folge davon

Anforderungsprobleme (Probleme im Problemraum)

Freie Universität Berlin

- Problem: Herausfinden, **was** genau gebaut werden soll

- Teilprobleme:
  - Problemdomänen sind komplex
  - Fachexperten können Bedarf nicht gut genug **ausdrücken**
    - (bei sehr innovativen Anwendungen gibt es gar keine Fachexperten)
  - und haben zuwenig **Fantasie**, sich SW-Möglichkeiten auszumalen
  - Verschiedene Gruppen von Fachexperten sind nötig.
    Sie bringen **widersprüchliche Anforderungen** ein
    - z.B. weil es Interessenkonflikte gibt
  - Fachexperten und Technikexperten benutzen verschiedene, manchmal inkompatible **Terminologie** und sehr verschiedene **Darstellungsformen**
    - und verstehen einander nur sehr schwer
  - Anforderungen **ändern sich** im Laufe der Zeit
    - oft schon lange vor Abschluss des Entwicklungsprojekts

Entwurfsprobleme (Probleme im Lösungsraum)

- Problem: Herausfinden, wie die SW strukturiert werden sollte, um die Anforderung gut zu erfüllen
  - Es gibt viele Möglichkeiten mit verschiedenen Stärken und Schwächen
  - Man muss darunter die günstigen entdecken und erkennen

- Teilprobleme:
  - Lösungsmöglichkeiten (er)kennen
  - Inakzeptable herausfiltern
  - Wirkung der Möglichkeiten auf die Qualitätseigenschaften verstehen
  - Prioritäten der Qualitätseigenschaften verstehen
  - Prioritäten gegeneinander abwägen ("Äpfel und Birnen")
    - Meist geht das nur per Umrechnung in Kosten
    - Die ist aber sehr oft dubios und sehr künstlich

Taxonomie
"Die Welt der Softwaretechnik"

Freie Universität Berlin

Welt der Problemstellungen:

- Produkt (Komplexitätsprob.)
  - Anforderungen (Problemraum)
  - Entwurf (Lösungsraum)
- **Prozess (psycho-soziale P.)**
  - **Kognitive Beschränkungen**
  - **Mängel der Urteilskraft**
  - **Kommunikation, Koordination**
  - **Gruppendynamik**
  - **Verborgene Ziele**
  - **Fehler**

Welt der Lösungsansätze:

- Technische Ansätze ("hart")
  - Abstraktion
  - Wiederverwendung
  - Automatisierung
- Methodische Ansätze ("weich")
  - Anforderungsermittlung
  - Entwurf
  - Qualitätssicherung
  - Projektmanagement

Prozessprobleme (psycho-soziale Probleme)

Freie Universität Berlin

Entstehen aus zwei Quellen:

- Menschen haben Schwächen und verhalten sich "merkwürdig"
- Zur SW-Konstruktion müssen Menschen zusammenarbeiten

- Diese Probleme entstehen aus dem Prozess heraus
  - und hängen nur wenig direkt vom Produkt ab

Kognitive Beschränkungen

- Die Denkfähigkeiten eines Menschen sind begrenzt

- Wichtigste Einschränkung:
  - Limitiertes Arbeitsgedächtnis (Kurzzeitgedächtnis plus Aufmerksamkeitslenkung):
  - es kann nur ca. 7±2 Elemente (*chunks*, Bündel) aufnehmen

Mängel der Urteilskraft

- Problem: Oft kann keiner der Beteiligten in einem Projekt eine Situation gut genug beurteilen, um eine fällige Entscheidung richtig zu treffen

Tritt vor allem in zwei Situationen auf:

- Radikales Vorgehen (technischer Bereich oder Prozess):
  - Emergente Eigenschaften lassen sich schwer abschätzen

- Rückgängigmachen falscher Entscheidungen (Prozessbereich):
  - Hier stehen psychologische Hürden im Weg (Verzerrungen der Wahrnehmung)

Kommunikation, Koordination

- Problem: Es ist schwierig, vorhandene Information stets korrekt und rechtzeitig an die Person weiterzugeben, die sie braucht
  - Inhaltliche Information (Anforderungen, Entwurfsentscheidungen etc.)
  - Organisatorische Information (Abstimmung, Koordination)

- Hauptquellen:
  - kognitive Beschränkungen oder Mängel der Urteilskraft
  - soziale Phänomene wie individuelle Abneigungen, z.B.
    - gegen Kommunikation allgemein (Introversion),
    - gegen bestimmte Personen (Aversion),
    - gegen die Kommunikation bestimmter Arten von Informationen (Überbringung schlechter Nachrichten)
    - usw.

Gruppendynamik

- Problem: Arbeiten in einer Gruppe beeinflusst Haltungen und Entscheidungen oft in unvernünftiger Art und Weise

- 2 Beispiele:
  - *"Groupthink"* führt dazu, dass gewisse (notwendige) kritische Fragen nicht mehr gestellt werden
    - z.B. bei Vorherrschen der Ansicht
      "Die automatisierten Tests müssen 100% des Codes abdecken"
  - *Selbsterfüllende Prophezeihungen* sind in Gruppen besonders wirksam
    - z.B. "Wir überziehen unseren Liefertermin immer."

Verborgene Ziele (hidden agendas)

- Problem: Menschen handeln nicht immer nur im Interesse des Projekts, sondern haben auch persönliche (meist unausgesprochene) Ziele
  - die *zum Teil* den Interessen des Projekts zuwider laufen

- Beispiele für verborgene Ziele
  - Selbst viel lernen
  - Spaß bei der Arbeit haben
  - möglichst elegante Lösungen finden (auf die man stolz ist)
  - Achtung bei Kollegen finden; Achtung bei Vorgesetzten finden
  - Gehaltserhöhung bekommen
  - früh nach Hause gehen; faul sein
  - spät nach Hause gehen; Privatleben verdrängen
  - bestimmte Kollegen schlecht aussehen lassen
  - bestimmten Kollegen aus dem Weg gehen
  - u.a.m.

Fehler

- Problem: Will ein Mensch X tun, so tut er oft Y
  - ohne es zu wollen oder zu merken und
  - ohne dass Y ein brauchbarer Ersatz für X wäre
    - Man nennt die Wahl von Y dann einen *Fehler*

```
//swap a and b:
b = a;
a = b;
```

- Resultat von Fehlern sind oft Defekte in Artefakten
- Resultate von Defekten ist oft ein Versagen der SW

- Fehler sind nur Erscheinungsformen der oben schon genannten Probleme
  - sind aber häufig genug und wichtig genug, um eine eigene Erwähnung zu rechtfertigen

Taxonomie
"Die Welt der Softwaretechnik"

Freie Universität Berlin

Welt der Problemstellungen:

- Produkt (Komplexitätsprob.)
  - Anforderungen (Problemraum)
  - Entwurf (Lösungsraum)
- Prozess (psycho-soziale P.)
  - Kognitive Beschränkungen
  - Mängel der Urteilskraft
  - Kommunikation, Koordination
  - Gruppendynamik
  - Verborgene Ziele
  - Fehler

Welt der Lösungsansätze:

- **Technische Ansätze ("hart")**
  - **Abstraktion**
  - **Wiederverwendung**
  - **Automatisierung**
- Methodische Ansätze ("weich")
  - Anforderungsermittlung
  - Entwurf
  - Qualitätssicherung
  - Projektmanagement

Welt der Lösungsansätze

- Was haben SW-Ingenieure und Softwaretechnik-Forscher(innen) an Ideen entwickelt, um den Problemen zu begegnen?

- Wir besprechen hier nur grundlegende Ideen
- Detaillierung ist Gegenstand des restlichen Semesters

Technische Ansätze ("hart")

- Diejenigen Ansätze, die sich weit genug konkretisieren lassen, um in Notationen oder Werkzeuge überführt zu werden

- Durch diese Konkretheit ("hartes" Wissen) sind technische Ansätze einfacher zu verstehen und anzuwenden als die "weichen" methodischen Ansätze
  - Methoden bieten aber ggf. mehr Anleitung für ihre Anwendung

- Anmerkung: Wir benennen im Folgenden nicht einzelne Ansätze, sondern Kategorien von Ansätzen
  - Die Kategorien selbst sind *nicht* sehr konkret
  - Die einzelnen technischen Ansätze sind es dann aber sehr wohl
    - Eigentlich sollte es besser heißen Abstraktion**en**, Wiederverwendung**en**, Automatisierung**en**

Abstraktion

- Abstraktion: Konzentration auf wesentliche Eigenschaften durch Weglassen von Details
  - Andere Sicht: Gruppierung gleichartiger Dinge gemäß einer Gemeinsamkeit und Ignorieren der sonstigen Unterschiede
  - Vorgehensweise: Bildung eines Konzepts (Begriffs) und Benennung dieses Konzepts durch eine Bezeichnung
- Kernprinzip der gesamten Informatik!
- Verwendung für alle möglichen Zwecke
  - z.B. Beschreibung von Produkten/Produktteilen, Vorgehensweisen, konkreten Vorgängen etc.
- Arten von Begriffen:
  - Allgemeine Begriffe (z.B. Elemente von Notationen wie etwa die Konstrukte einer Programmiersprache)
  - Vorhabensspezif. Begriffe (z.B. einzelne Module eines Systems)
    - Begriff/Abstraktion reduziert den Baustein auf seine Schnittstelle
    - Modulname ist die Bezeichnung der Abstraktion

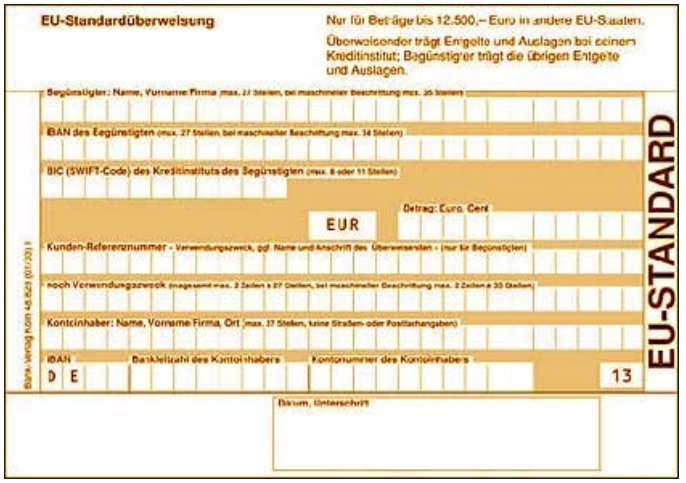
Eine Abstraktion

Wiederverwendung

- Spezialfall von Abstraktion:
  - Benutze nicht nur den Begriff mehrmals, sondern auch damit verbundene Details

- Wiederverwendung geht nicht nur bei Programmcode, sondern auch für z.B.
  - **Arbeitsprodukte** wie Anforderungen, Anforderungsmuster, Architekturen, Teilentwürfe, Entwurfsmuster, Testfälle
  - **Prozesshilfsmittel** wie Dokumentschablonen, Vorgehensbeschreibungen, Checklisten, Prozessmuster, Softwarewerkzeuge

- **Wiederverwendung ist die Hauptquelle von Produktivitäts-verbesserungen in der Softwaretechnik!**



Automatisierung

- Übertrage eine wiederholt zu erledigende Tätigkeit dem Computer
  - spart Zeit (und damit Kosten)
  - vermeidet triviale Durchführungsfehler
- Kann als Spezialfall von Wiederverwendung betrachtet werden
- Ist das Automatisierungsprogramm flexibel
  für viele Situationen einsetzbar,
  so nennt man es **Softwarewerkzeug**
  - spezialisierte, weniger flexible heißen meist **Skript**

Taxonomie
"Die Welt der Softwaretechnik"

Freie Universität Berlin

Welt der Problemstellungen:

- Produkt (Komplexitätsprob.)
  - Anforderungen (Problemraum)
  - Entwurf (Lösungsraum)
- Prozess (psycho-soziale P.)
  - Kognitive Beschränkungen
  - Mängel der Urteilskraft
  - Kommunikation, Koordination
  - Gruppendynamik
  - Verborgene Ziele
  - Fehler

Welt der Lösungsansätze:

- Technische Ansätze ("hart")
  - Abstraktion
  - Wiederverwendung
  - Automatisierung
- **Methodische Ansätze ("weich")**
  - **Anforderungsermittlung**
  - **Entwurf**
  - **Qualitätssicherung**
  - **Projektmanagement**

Methodische Ansätze ("weich")

- Methodische Ansätze sind nur schwach vorstrukturiert
  - deshalb als "weich" empfunden (kein "hartes Wissen")
  - Benötigen menschliche Intelligenz zur Durchführung
    - Echte "Methoden" (im Sinne von Kochrezepten) sind in der Softwaretechnik illusorische Idealbilder!
    - Hilfreiche Methodenelemente oder Praktiken gibt es aber sehr wohl

- Abgrenzung zu technischen Ansätzen:
  - Kein Teil des Problems lässt sich abspalten und universell lösen
    - (alles, was sich doch abspalten und universell lösen lässt, zählen wir also nicht mehr zu den methodischen Problemen)
  - Die Übergänge sind aber dennoch fließend

Anforderungsermittlung

- Einsicht: Man darf sich nicht auf intuitiven Eindruck darüber verlassen, was gebaut werden sollte
  - sondern sollte die Anforderungen systematisch ermitteln
- Prinzipien:
  - **Erhebung** der Anforderungen bei allen Gruppen von Beteiligten
  - **Beschreibung** in einer Form, die die Beteiligten verstehen
  - **Validierung** anhand der verschriftlichten Form
  - **Spezifikation**: Übertragung in eine zur Weiterverarbeitung günstige Form
  - **Trennung von Belangen**: Anford. kopplungsarm ausdrücken
  - **Analyse auf Vollständigkeit**: Lücken aufdecken und schließen
  - **Analyse auf Konsistenz**: Widersprüche aufdecken und lösen
  - **Mediation**: Widersprüche, die auf Interessengegensätzen beruhen, einer Lösung zuführen (Kompromiss oder Win-Win)
  - **Verwaltung**: Übermäßige Anforderungsänderungen eindämmen, Anforderungsdokument immer aktuell halten.

Entwurf

- Einsicht: Besser nicht einfach drauf los programmieren,
  - sondern Struktur u. Funktionsweise vorab im Ganzen festlegen,
  - um bessere Qualität zu ermöglichen und
  - um arbeitsteilige Realisierung zu erleichtern
- Prinzipien:
  - **Trennung von Belangen (separation of concerns)**: Belange (insbes. Funktionen) so von einander trennen, dass man sie einzeln verstehen, realisieren und verändern kann
  - **Architektur**: Treffe globale Festlegungen für die Gestaltung nicht abtrennbarer Belange (insbesondere: nichtfunktionale Anforderungen)
  - **Modularisierung**: Verberge die Umsetzung von Belangen möglichst hinter einer **Schnittstelle** (**information hiding**)
    - dadurch wird der Belang **lokal** und ist einfacher zu ändern
  - **Dokumentation**: Entwurf u. Entwurfsentscheidungen schriftlich festhalten (jeweils: Zweck, Alternativen, Argumentation)

Qualitätssicherung

- Einsicht: Man macht oft Fehler, die zu schweren Mängeln in der SW führen können
  - Solchen Mängeln sollte man vorbeugen

Prinzipien:

- Konstruktive Qualitätssicherung
  - Ergreife **vorbeugende Maßnahmen**, die vermeiden helfen sollen, dass überhaupt erst etwas falsch gemacht wird
  - Oft auch genannt **Qualitätsmanagement** (evtl. inkl. prüfende Maßnahmen) oder **Prozessmanagement**
- Analytische Qualitätssicherung
  - Verlasse dich nie ganz auf die Vorbeugung, sondern verwende zusätzlich **prüfende Maßnahmen**, die entstandene Mängel aufdecken sollen, damit man sie beheben kann
  - Wichtigste Vertreter (bislang): **Test, Durchsichten**

Projektmanagement

- Einsicht: Bau einer größeren SW verlangt nach Planung, Leitung und Koordination
- Prinzipien:
  - **Zielsetzung**, Prioritäten, Pläne, individ. Aufgaben müssen allen Beteiligten klar sein und sie sollten sich damit identifizieren
  - **Stabile Anforderungen**: A. dürfen s. nicht zu schnell verändern
  - **Iteration**: Projekt muss in kurzen Abständen wohldefinierte Ergebnisse hervorbringen
  - **Kommunikation**: Informationen müssen rechtzeitig die richtige(n) Person(en) erreichen
  - **Konflikte** zw. Beteiligten auf zielführende Weise lösen
  - **Risikomanagement**: Identifiziere wichtige unerwünschte Ereignisse, leite vorbeugende Maßnahmen ein oder bereite Gegenmaßnahmen vor; überprüfe regelmäßig
  - **Normales Vorgehen**: Vermeide radikales Vorgehen, es sei denn, der erwartete Nutzen rechtfertigt es

Anmerkung zur Taxonomie

- Die Taxonomie ist alles andere als kanonisch
  - man kann über vieles darin trefflich streiten
- aber sie liefert einen nützlichen Orientierungsrahmen

- Bitte prüfen Sie am Ende der Vorlesung anhand der Taxonomie Ihr Wissen
  - Es sollte Ihnen zu jedem Eintrag einiges aus der Vorlesung einfallen
    - und sie sollten jeden Teil der Vorlesung prompt in der Taxonomie einordnen können
  - Sie sollten Einordnung und Wichtigkeit jedes Eintrags charakterisieren können
  - Es gibt zu diesem Zweck ein **Mini-Skript**, in dem zur Taxonomie etwas mehr steht als auf den Folien

...und bitte nie vergessen:

- Verwenden Sie stets Ihren gesunden Menschenverstand!

Danke!