UTGÅNGSPUNKT

'Programmering' (= kodning) och design (= konstruktion) är teknikområden.

Framställning av stora/komplexa system kräver dessutom t ex

- många programmerare/teams
- personalfrågor (specialister, utbildning, ersättare ...)
- externa frågor (marknadsföring, kontrakt ...)
- kvalitetssäkring (produkter, processer ...)
- dokumentation, olika slag
- management (ledning, uppföljning, resursfördelning ...)

Till stor del icke-tekniska frågeställningar.

Området Software Engineering omfattar både dessa tekniska och icke-tekniska kompetenser.

Metodiskt arbetssätt i projektarbetsform behövs. Allmänt:

1 förstå problemet
2 planlägg lösningen
3 genomför planen
4 utvärdera resultatet

KURSMÅL:

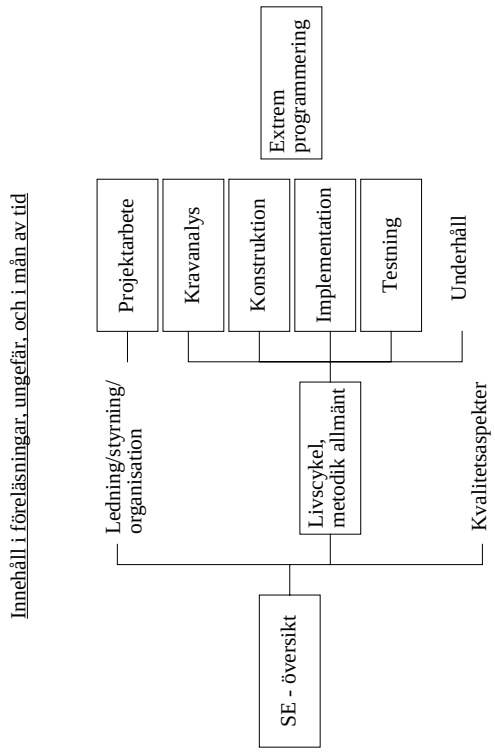
- Kodning av lite *större volym*
- Programutveckling i *metodisk projektform*, med dokumentationer
- Kunskap om *några element* inom Software Engineering

FÖRKUNSKAPER:

- God förtrogenhet med något 'högnivåspråk'
- Praktisk kunskap om datastrukturer och algoritmer

EXAMINATION:

- Hemtentamen på SE-relaterat stoff
  - D Bell: "Software Engineering for Students - A Programming Approach" Addison-Wesley 2005
  - Valfria källor
- Projektgenomförande, med
  - rudimentär kravspecifikation
  - designspecifikation och -presentation
  - implementation
  - användarmanual
  - efterstudie
- Endast betyg G



SOFTWARE ENGINEERING ?

En praktisk och vetenskaplig del av datalogi (?)
- som anger metoder, verktyg, riktlinjer, attityder ...
- för konstruktion av (stora, komplexa) PV-system ...
- i enlighet med användares/beställares intentioner ...
- inom föreskrivna budget- och tidsramar ...
- och med hänsyn till kvalitets- underhållsaspekter ...

> "SE is the body of THEORY and PRACTICAL TECHNIQUES that can be brought to bear on the process of developing [... *som syftar till att kunna utveckla* ...] software."

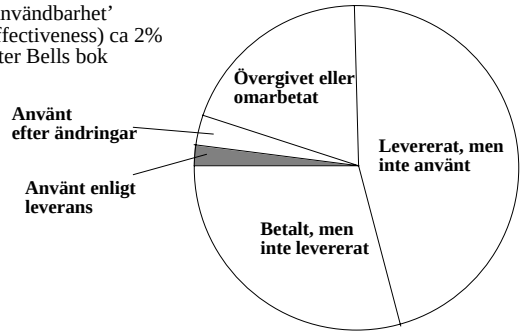
MEST ÖVERGRIPANDE MÅL ?

- förmåga att urskilja och rätta sig efter kunders önskemål/krav (behovsstyrt, inte teknikstyrt!)
- användande av ingenjörsmässiga principer/ idéer/kvaliteter/attityder

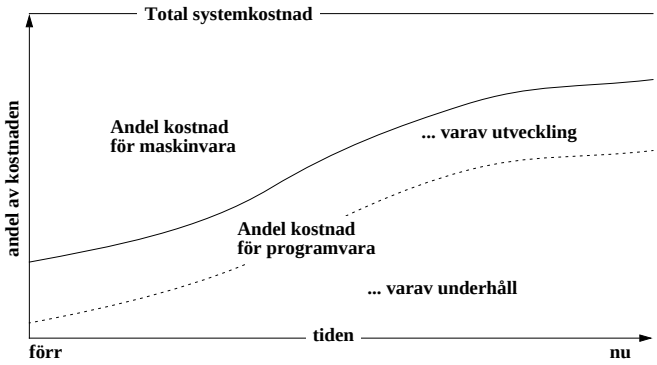
INGENJÖRSMÄSSIGHET ?

> Konstruktion av PV är inte (längre) ett 'ad hoc'-jobb utfört av enstaka, kreativa individer i den mörka skrubben, utan ett välorganiserat, metodbaserat teamwork, baserat på känd teknik.

‘Användbarhet’
(effectiveness) ca 2%
Efter Bells bok



VARFÖR SE UPPKOMMIT



TRADITIONELL ARBETSGÅNG, GROVT

Projektfas, allmänt	SE-fas	produkt/ dokument
1. *förstå problemet*	kravanalys	kravspecifikation
2: *planlägg lösningen*	planering (tid, resurser)	projektplan
3: *genomför planen*	design (konstruktion)	designspec., ev på flera nivåer
	implementation (kodning)	kod
4: *utvärdera resultatet*	testning (validering, verifikation) flera nivåer	ny kod, uppdaterade dokument

Nedbrytning i flera steg, mer precisierade delsteg.

PROJEKT ?

> “En tillfällig kraftsamling (endeavour) som genomförs för att skapa en unik produkt, tjänst eller resultat.” (efter PM-BOK 2004)

- ett *definierbart* ändamål:
  Det definieras i en kravspecifikation - funktionalitet, prestanda, uppträdande ...
- ett *unikt* företag:
  Inte ‘rutinarbete’, avser inte nåt som gjorts identiskt tidigare
- en *tillfällig* aktivitet:
  Givna tidpunkter för start och avslut, enligt krav.

Annan definition:

> “Ett projekt är en kombination av resurser som förs ihop för att skapa något som inte fanns förut.” (efter Cleland och Ireland, 2002)

PROJEKT’HANTERING’ (management) ?

planering, organisering, styrning, övervakning, förändring, hantering av osäkerhet, risker, ...

Speciellt för PV-projekt:
- produkten är icke-konkret, ‘osynlig’, svårt för en manager att urskilja framstegen
- standardiserade processer saknas, det går inte att förutsäga när problem kommer att uppstå
- den snabba teknikutvecklingen kan orsaka att tidigare erfarenheter inte längre gäller

BEGREPPSDISTINKTIONER

Metod: ett specifikt, konkret, detaljerat, strukturerat förfaringssätt inklusive verktyg. ‘Algoritmisk’.

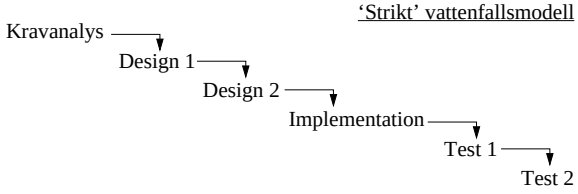
t ex JSP, XP

Metodik: grundmodeller, generella och gemensamma drag, för en räcka metoder. ‘Processmodell’ ?

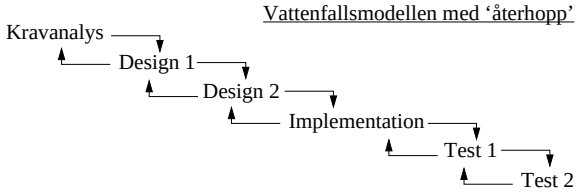
t ex prototyping, unified process

Metodologi: ‘läran om’ hur metoder konstrueras, kan värderas, generella egenskaper

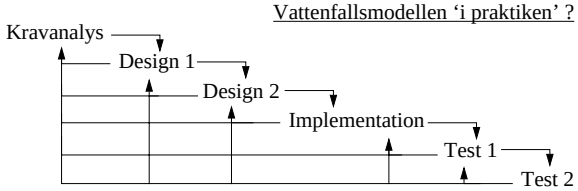
‘Strikt’ vattenfallsmodell



Vattenfallsmodellen med ‘återhopp’



Vattenfallsmodellen ‘i praktiken’ ?



FLER BEGREPP: V & V

VALIDERING:

Med VALIDERING menar man att slå fast att det som levereras (eller interna dokument på vägen) har en FUNKTIONALITET i enlighet med det beställaren avsett. Det relaterar till KUNDEN, hans förväntningar

VERIFIKATION:

Med VERIFIERING menar man, något striktare, att slå fast att produkten (och steg på vägen!) lever upp till kravspecifikationen! Denna kan ju beskriva mer än funktionaliteten (andra kvaliteter). Verifiering kan också avse koll av mer interna tekniska lösningar, allmänna programvarukrav, ...

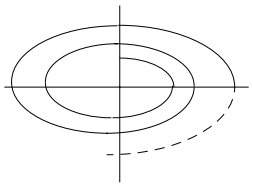
Lite POPULÄRT har det sagts (Boehm):

VALIDATION: Are we building the right product?

VERIFICATION: Are we building the product right?

SPIRALMODELLEN (Boehm 1988)

Varje varv utgör en ‘fas’.
Exempelvis ...



v 1: ger en ‘concept of operation’
v 2: ger en validerad kravspec
v 3: ger en validerad design
v 4: ger en testplan
v 5: ...

I varje varv äger i princip samma följd av delaktiviteter rum, oavsett fas. Kan i korthet vara t ex ...

Sätt upp målet:
Beskriv målet för ‘varvet’, identifiera begränsningar, risker, konstatera vilka alternativ som kan väljas.
Riskanalys:
Värdera alternativ, välj bästa utifrån befintliga risker.
Utveckling, verifiering/validering:
Genomför aktiviteten. Kolla.
Planering:
Utvärdering, beslut om fortsättning, planering inför den.

FRAMSTÄLLNING AV KRAV (Requirements engineering)

1 Rimlighetsstudie (feasibility study)
Kan systemet bidra med något? ... implementeras? ... integreras?

2 Kravinsamling (elicitation) och analys
Svårt: Verksamhetsfolk (VF) kanske inte kan uttrycka det dom vill, är orealistiska, har ‘egna’ och varierande begrepp, outtalad kunskap. Verksamhetens omgivning kan förändras.

Procedur:
i) Förstå verksamhetens domän
ii) Samla krav från VF
- vy-orienterad metod
- scenarier
- ‘etnografisk’ metod

iii) Klassificera krav
iv) Lös konflikter mellan krav
v) Prioritering mellan krav
vi) Kolla kravens konsistens, kolla mot kunden

Detta sker förstås iterativt, och vissa aktiviteter i samarbete med VF.

3 Kravspecifikation (processen)
Skriv ihop dokumentet.

4 Kravvalidering
Kolla giltighet, konsistens, kompletthet, rimlighet, verifierbarhet ...

Några egenskaper hos en kravspecifikation

Den ..

- ska specificera bara det externa uppträdandet (vad, inte hur)
- ska speca eventuella begränsningar i implementationen (men inte många!)
- ska själv vara lätt att ändra (för detta kommer att behövas!)
- ska kunna funka som referens vid underhåll, och under alla utvecklingsfaser
- ska också klargöra vilka anständiga responser som ska ges på oönskade händelser under drift.

En kravspecifikation kan innehålla ...

- Identifikation
- Sammanfattning
- Innehållsförteckning
- Inledning
- Användarna
- Funktionella krav
  - skall-krav
  - bör-krav
  - 'eventuellt'-krav
- Produktkomponenter
  - programvara
  - maskinvara
  - dokumentation
- Effektivitet
- Kompatibilitet
- Konfiguration
- Installation och service
- Tillförlitlighet
- Ordförklaringar
- Index

Exempel på rubrikerna i en 'verklig' kravspecifikation

1 Revisionshistoria
2 Introduktion
2.1 Syfte
3 Definitioner och förkortningar
4 Projektets syfte
4.1 Projektbakgrund
4.2 Mål
4.3 Effekt
5 Kund och andra intressenter
5.1 Kund
5.2 Andra intressenter
6 Användare
7 Avgränsningar
7.1 Lösning
7.2 Implementation
7.3 Externa kopplingar
7.4 Övrigt
8 Fakta och förutsättningar
9 Krav
9.1 Händelseflöden
9.2 Funktionalitet
9.3 Användargränssnitt
9.4 Användbarhet
9.5 Felhantering
9.6 Data
9.7 Statistik och rapporter
9.8 Prestanda
9.9 Säkerhet
9.10 Administration och informationsförsörjning
10 Osäkerheter och risker
10.1 Beroenden
10.2 Stabilitet
10.3 Prestanda
10.4 Andra identifierade risker
11 Användardokumentation och utbildning
12 Tidsuppskattning
13 Väntrum
14 Lösningsidéer

Projektarbete kräver ...

- fördelning av arbete mellan grupper, individer, ev. med hänsyn till kompetenser
- flitig kommunikation mellan grupper/individer
- beroende mellan individers/gruppers alster
- en tidsplan
- kontrollpunkter för att avgöra om planen måste ändras
- dokumentation, för att
  - beskriva mål och förlopp för ev. ersättare
  - samla erfarenheter
  - förbättra processen ...

En projektplan skall (åtminstone) ...

- vara nedskriven i förväg
- beskriva vad som ska uträttas (inte hur)
- vara välskriven, strukturerad, kortfattad, begriplig
- ta hänsyn till tänkbara 'olyckshändelser'
- gärna vara modulariserad

En projektplan kan innehålla

- Översikt
  Intro till jobbet, kunden. Planens egen org.
- Fasplan
  Vilka utvecklingsfaser (metoder), produkter, datum?
- Organisationsplan
  Vilka team, ansvar?
- Testplan
  Vem? Procedurer? Verktyg?
- Förändringsplan
  Hur hantera?
- Dokumentationsplan
  Vilka, när, till vem? Vem godkänner?
- Utbildningsplan
  Internt, externt. Vem, när, resurser?
- Plan för rapportering och 'reviews'
  Vad, till vem, när?
- Installationsplan
  Vilken procedur?
- Plan för kvalitetssäkring
  Standarder?
- 'Varuplan' (.. deliverables)
  Vad ska levereras, när?
- Resursplan
  Persontid, datortid. Summering av milstolpar.

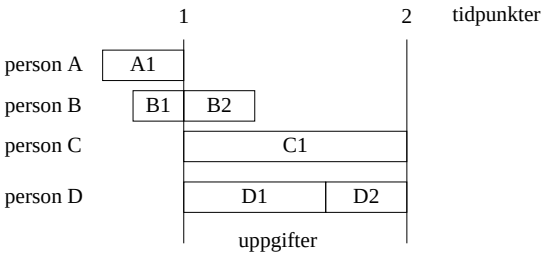
EXEMPEL PÅ MILSTOLPAR
från ett 'PUM-projekt':

Fas	Milstolpe	Datum när milstolpen planeras uppnås
Förstudiefasen	Förstudiedokument klart	940205
Definitionsfasen	Kravspecifikationsdokument klart	940225
	Kontrakt skrivet	940225
	Acceptansvillkor framställda	940225
	Projektplan klar	940304
Designfasen	Designdokument klart	940415
	Programmerarhandbok klar	940413
	Systemtestfall framställda	940405
	Integrationstestfall framställda	940408
	Modultestinstruktioner framställda	940415
Programmeringsfasen	Teknisk dokumentation klar	940506
	Användarhandledning klar	940506
Testfasen	Modultest klara	940422
	Integrationstest klar	940503
	Systemtestning klar	940506
Avslutningsfasen	Efterstudiedokument klart	940520
	Acceptansöverenskommelse klar	940510
	Slutrapport från kvalitetsarbetet klar	940520
	Återkoppling klar	940520

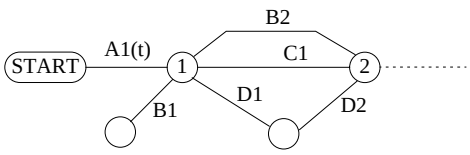
ANM: För täta, för 'små.
Borde nog innehålla mer av 'godkänt/accepterat'.

PLANERINGSVERKTYG:

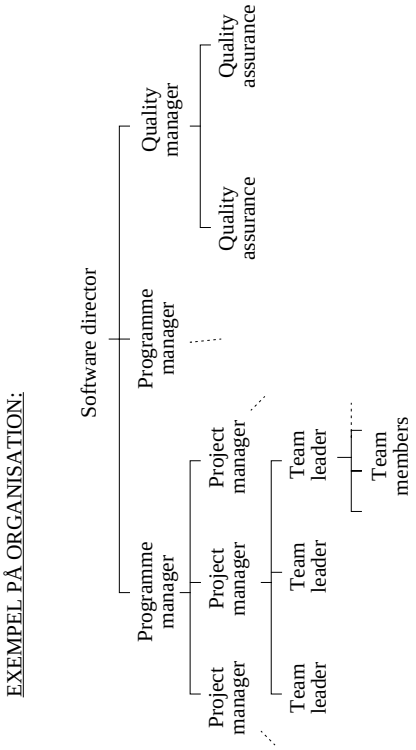
'Bar Chart' - GANTT-diagram



'Activity network' - PERT-diagram



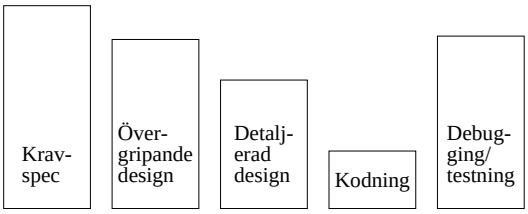
EXEMPEL PÅ ORGANISATION:



KOMPETENSKRAV ?

Graden av ’skicklighet’ som kan krävas för olika faser eller arbetsmoment vid programvaruutveckling.

OBS! Taget från Bells lärobok !



Exempel på rubrikerna i ’verklig’ projektplan (1)

1 Revisionshistoria
  1.1 Ändringslogg
  1.2 Relaterade dokument
2 Förutsättningar och bakgrund
  2.1 Syfte
  2.2 Bakgrund
3 Mål
  3.1 Affärsmål
  3.2 Systemmål
  3.3 Kvalitetsmål
4 Omfattning och resultat
  4.1 Projektets uppgift
  4.2 Avgränsningar
  4.3 Förväntade resultat
5 Kopplingar till andra projekt

6 Projektorganisation
  6.1 Styrgrupp
  6.2 Projektorganisation
  6.3 Ansvar
  6.4 Ansvarmässig avgränsning
  6.5 Projektmöten
  6.6 Samverkan och rapportering
  6.7 Resursplan
7 Arbetsmetodik
  7.1 Arbetsmetod
  7.2 Verifiering
  7.3 Validering
  7.4 Upphandling/köp
  7.5 Kommunikationsplan
  7.6 Testplan
  7.7 Kvalitetsplan

Exempel på rubrikerna i ’verklig’ projektplan (2)

8 Tidplan och milstolpar
  8.1 Milstolpar
  8.2 Tidplan
  8.3 Leverabler från kund
  8.4 Utrullning
  8.5 Kriterier för överlämning

9 Kostnader
  9.1 Utvecklingsmiljö
  9.2 Testmiljö
  9.3 Produktionsmiljö
10 Risker
  10.1 Beroenden
  10.2 Stabilitet
  10.3 Prestanda
  10.4 Andra identifierade risker

11 Projektavslut
  11.1 Överlämning till drift och förvaltning
  11.2 Utvärdering
12 Ändringshantering

DESIGN - NÅGRA VIKTIGA NYCKELORD

- modularisering
  = *små, överblickbara bitar*
- modulkvalitet
  = *’bra’ egenskaper som kännetecknar dessa*
- information hiding
  = *göm viss, detaljerad ’kunskap’*
- abstraktioner (en följd av info hiding)
  = *access via gränssnitt*
- hierarkisk syn
  = *lagermodell*
- språkoberoende
  = *’högre’ nivå, mer oberoende implementation*
- metodik
  = *hur man urskiljer moduler*

Modul

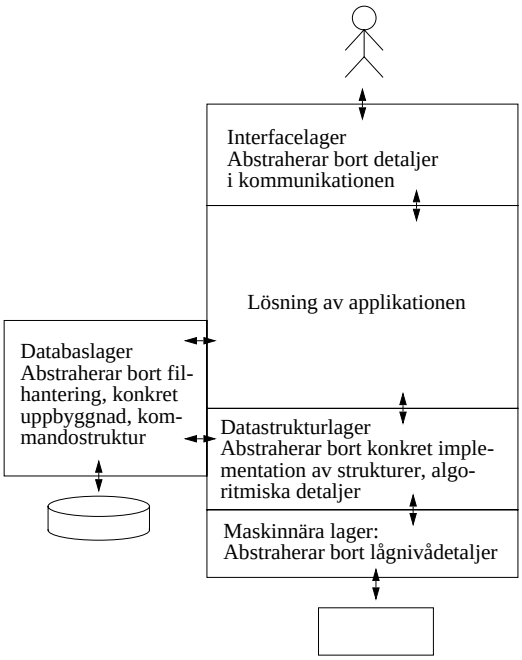
Viktigast: överblickbar, förståelig för en person/ett team.
Därför ska den

- inte vara alltför stor (eller komplex)
- ha ett väl avgränsat syfte (~ hög ‘cohesion’)
- ha få relationer till andra moduler (= låg ‘coupling’)

En modul kan vara t ex

- en enda subrutin (eller process)
- en databeskrivning (utan algoritmer)
- (oftare?) ett hopbygge av flera, mindre sådana, om hög cohesion råder. En klass (motsvarande)!

Modul - abstraktioner - lager



KWIC-index (efter van Vliet)

KWIC = Key Word In Context

Ursprunglig text:

Software Engineering should be a compulsory topic.

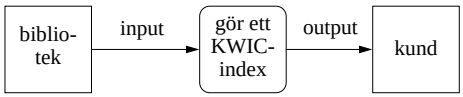
Generera alla ’ordvisa’ cirkulära skift:

Software Engineering should be a compulsory topic. Engineering should be a compulsory topic. Software should be a compulsory topic. Software Engineering be a compulsory topic. Software Engineering should a compulsory topic. Software Engineering should be compulsory topic. Software Engineering should be a topic. Software Engineering should be a compulsory

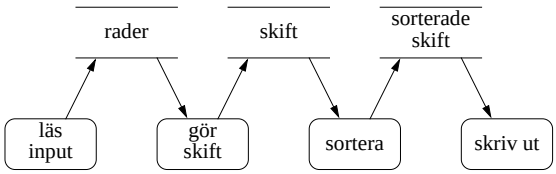
Sortera skiften alfabetiskt:

a compulsory topic. Software Engineering should be be a compulsory topic. Software Engineering should compulsory topic. Software Engineering should be a Engineering should be a compulsory topic. Software should be a compulsory topic. Software Engineering Software Engineering should be a compulsory topic. topic. Software Engineering should be a compulsory

Dataflödesdesign - notation

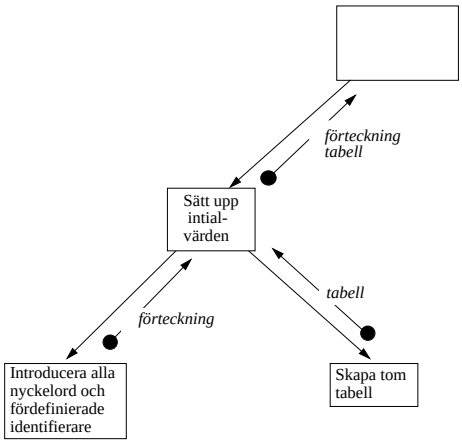


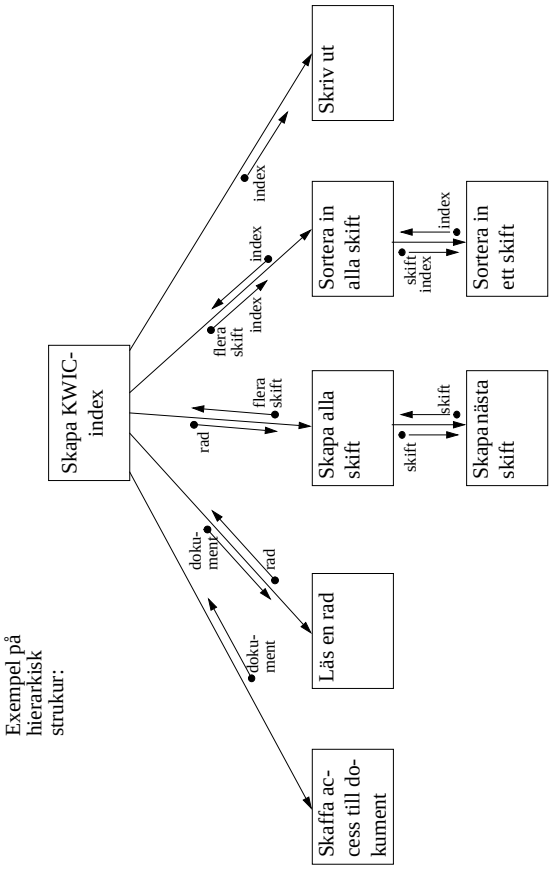
Kontextdiagram



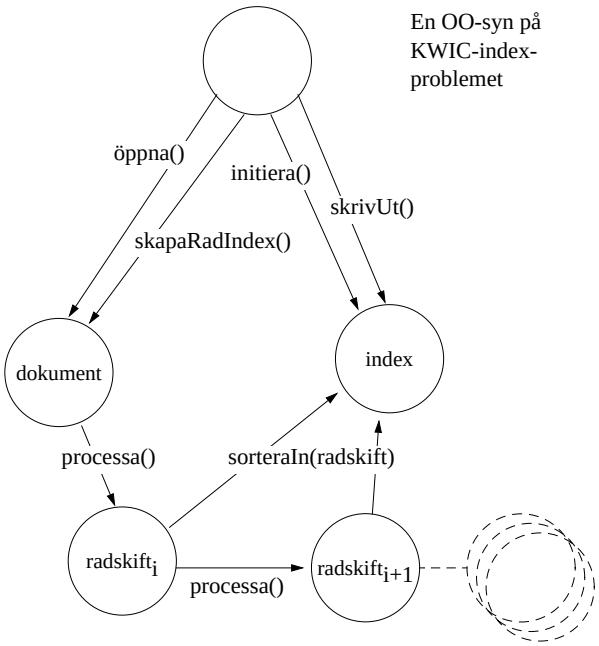
Dataflödesdiagram

Arkitektur (µ-skala!)





Exempel på hierarkisk strukur:



En OO-syn på KWIC-index-problemet

DETALJERAD DESIGN
(MODULDESIGN, MODULKONTRAKT)

Förslag på modulattribut som måste specificeras:

(IEEE Standard 1016, från van Vliet)

1. Identifikationen (namnet), unikt
2. Typen, alltså t ex subsystem, package, klass, fil, subrutin, ...
3. Syfte, övergripande
4. Funktionen, relaterat till kravspecen
5. Beståndsdelar, om sådana i sin tur finns
6. Beroenden, relationer till andra komponenter
7. Gränssnitt till andra komponenter, i detalj
8. Externa resurser
9. Utförande, beskrivning av algoritmer, exceptions etc - en förfining av funktion. Motivera val!
10. Data, beskrivning av representation, format, avsikt med interna data

IMPLEMENTATION:

Begreppet *strukturerad programmering* kan ‘förklaras’ på olika sätt:

- “Programkod ska kunna läsas och förstås i den följd programtexten anger”
- “Använd styrstrukturerna på ett systematiskt sätt”
  En ingång, utgång
- “Texten ska återspegla den logiska strukturen”
- “Skriv för folk, inte för kompilatorer”
- “Använd inte GOTO”

EVALUERING - BEDÖMNING AV SPRÅK

Vilka möjligheter har man att i språket kunna

- modularisera på ett vettigt sätt, också i ‘det stora’?
- göra egna abstraktioner?
- införa information hiding?
- skapa oberoende mellan moduler?
- skriva läsbar kod?

Hur är det med

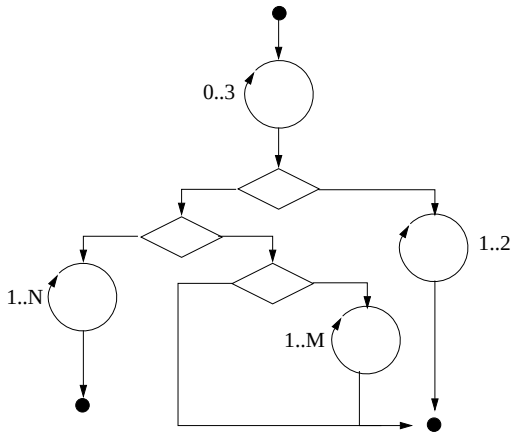
- ortogonaliteten?
  få, kraftfulla konstruktioner som kan kombineras godtyckligt
- stark typning?
  gäller ‘blandning’ av typer, typkontroll, när denna äger rum

UTNYTTJA DET SPRÅKET TILLHANDAHÅLLER!

Exempel på saker som kan ingå i en checklista vid walkthrough - inspection av kod:

- felaktig användning av data
  oinitierade variabler, arrayindex utanför gränser, “dangling pointers”
- deklarationsfel
  användning av icke deklarerade storheter, dubbel deklaration i ett block
- beräkningsfel
  0-division, overflow, typmismatch, operatorprioriteter
- logikfel
  < i stället för <=, and-or, prioriteter
- fel i styrning
  oändliga loopar, varvräknarfel
- interfacefel
  antalet, typerna/sorterna på parametrar, globala data

På hur många sätt går det att ta sig från start till slut här?



4 x ((N + (1 + M)) + 2) ??

Effektivitet av felsökning i ett realtidsprojekt med 400 utvecklare (van Vliet):

	%-andel funna designfel	%-andel funna kodningsfel	total-effektvitet
Designreview	54	-	54
Kodreview	33	84	64
Testning	38	38	38

Kostnadseffektivitet, samma studie:

Designreview	Kodreview	Testning
8.44	1.38	0.17

TESTNING - EXEMPEL (1):

En ‘specifikation’:

Ett program läser in tre heltalsvärden från en rad. De tre värdena tolkas som längderna på tre sidor i en triangel. Programmet skriver ut ett meddelande som anger om triangeln är likbent, liksidig eller om alla tre sidorna är olika.

Uppgift:

Skriv ett antal testfall (dvs ’kategorier’ plus några specifika testdata per kategori) som du anser testar programmet på ett bra sätt.

TESTNING - EXEMPEL (2):

Kod:

```
#include <stdlib.h>
#include <stdio.h>

int main(void) {
   int x, y, z;

   scanf("%d %d %d", &x, &y, &z);

   if (x == y && y == z)
      printf("Triangeln är liksidig.\n");
   else if (x == y || x == z || y == z)
      printf("Triangeln är likbent.\n");
   else
      printf("Alla sidor i triangeln är olika.\n");
   return EXIT_SUCCESS;
}
```


TESTNING - EXEMPEL (3):

Exekvering:

```
unix> make testprogram
.........


unix> testprogram
1 1 100
Triangeln är likbent.

unix> testprogram
0 0 0
Triangeln är liksidig.

unix> testprogram
-1 -1 -1
Triangeln är liksidig.

unix> testprogram
3.0 3.0 3.0
Alla sidor i triangeln är olika.

unix> testprogram
Klas, Kalle och Lotta
Alla sidor i triangeln är olika.
```

TESTNING - EXEMPEL (4):

‘Självevaluering’ angående testfall/-data:

1. Har du testfall som representerar en giltig oliksidig triangel! OBS att fall som 1, 2, 3 eller 2, 5, 10 inte gör det - sådana trianglar finns inte!

2. Har du testfall som representerar en giltig liksidig triangel?

3. Har du testfall som representerar en giltig likbent triangel? OBS att ett testfall som har 2, 2, 4 inte kan räknas!

4. Har du åtminstone tre testfall som representerar giltiga liksidiga trianglar, så att du provar alla tre permutationer av två lika sidor (t ex 3, 3, 4 och 3, 4, 3 och 4, 3, 3)?

5. Har du ett testfall där en sida är 0?

6. Har du ett testfall där en sida är negativ?

7. Har du ett testfall med tre heltal större än noll där summan av två är lika med det tredje? Dvs, om programmet påstår att 1, 2, 3 representerar en oliksidig triangel så innehåller det en bug!

8. Har du åtminstone tre testfall i kategori 7 så att att du provar alla permutationer där en sida är lika med summan av de andra två (t ex 1, 2, 3 och 1, 3, 2 och 3, 1, 2)

TESTNING - EXEMPEL (5):

‘Självevaluering’ ... -forts.

9. Har du ett testfall med tre heltal större än 0 så att summan av två är mindre än det tredje? (1, 2, 4 eller 12, 15, 30).

10. Har du testat minst tre permutationer av ovanstående?

11. Har du ett testfall där alla sidorna är 0?

12. Har du ett minst ett testfall med värden som inte är heltal?

13. Har du åtminstone ett testfall som specificerar fel antal värden?

14. *För varje testfall ovan, specificerade du (i förväg!) det förväntade resultatet?*

utarbeta och dokumentera tester, inkl förväntat utfall, i förväg !

gör alltså inga tester ’i flykten’ !

använd dina mest kompetenta (kreativa) programmerare som testare !

en test kan aldrig bevisa frånvaron av fel - bara påvisa förekomsten !

en test som inte påvisar något fel är förmodligen en undermålig test (, eller ... ?)

när ska man sluta testa ?

EXTREM PROGRAMMERING (1)

Extremt?

- moment genomförs kontinuerligt
- utvecklingscykler är små, korta
- lite dokumentation utanför koden

Vad krävs/önskas?

- inte för stora projekt
- kundrepresentant på plats
- planerings’spel’, ’user stories’
- helst ’parprogrammering’

Karakteristiskt?

- utgår från utvecklares behov
- ingen stor administrativ apparat, lite dokumentation
- smidig hantering av förändringar
- snabba resultat, god kvalitet förväntas
- ingen egentlig design-spec
- kanske inte heller sedvanlig kravspec!

Planering:

- täta möten kund - utvecklare
- planerings’spel’ - ’user stories’
- små, täta uppdateringar till kunden
- kund hos utvecklarna

EXTREM PROGRAMMERING (2)

Design:

- ingen specifikation i förväg
- enhetstester/testfall produceras före motsv. kod, är en form av spec. för enheten
- ofta automatiserat testande
- acceptanstest av varje ’story’
- ’refactoring’
  - byt namn på metod/något
  - gör en bit kod till egen metod/motsv.
  - flytta på metoder/...
  - verktyg finns, görs för ökad läsbarhet, förenklingar

Kodning:

- parprogrammering
- kollektivt ägande
- kontinuerlig integration
- ’fysisk närhet’ mellan utvecklare

Dokumentation:

- vissa kundkrav
- annars kräver xp inte mycket
- koden är viktigaste dokumentet, skall vara högkvalitativ
- ’story’na, diagram, ...
- acceptanstester kan fungera som kravspec.