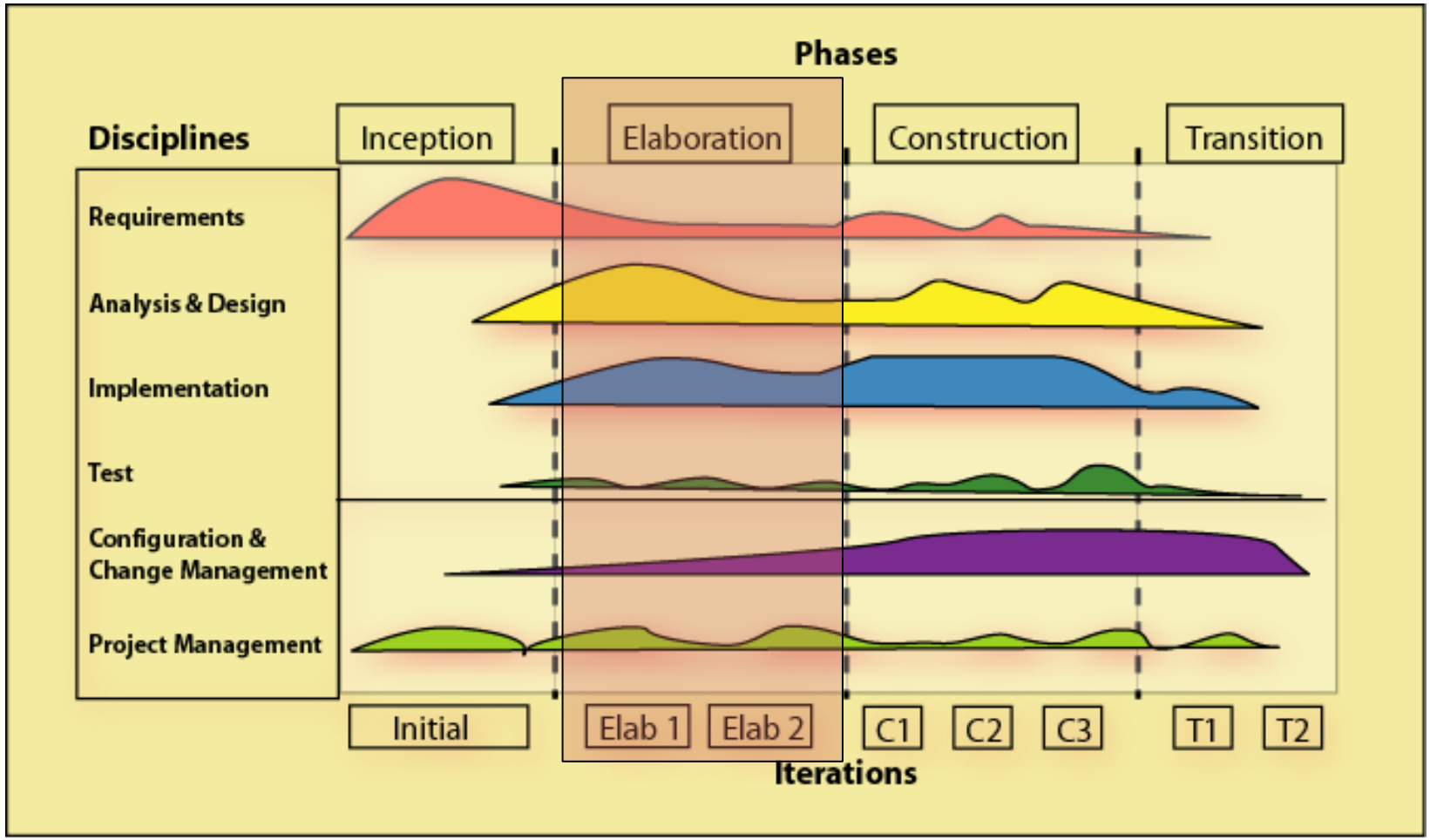
Elaboration

1DV404, HT14

Jesper Andersson

Kap **13, 33**, 34, 39

UP – Faser


Phases
Disciplines
Inception
Elaboration
Construction
Transition
Requirements
Analysis & Design
Implementation
Test
Configuration &
Change Management
Project Management
Initial
Elab 1
Elab 2
C1
C2
C3
T1
T2
Iterations

Elaboration

- ✓ **Syfte:** Fastställa en basarkitekturen för systemet vilket ger en stabil grund för den största delen av utvecklingsarbetet i nästa fas.

- ✓ Krav
  - Få en mer detaljerad förståelse av kraven.
  - Få en fördjupad förståelse för de kritiska krav som valideras av arkitekturen ➔ T. ex. förstå hur de samverkar
- ✓ Design
  - Designa, implementera, och validera ➔ fastställa basarkitekturen.
  - **Basarkitektur**: Ett skelettsystem, med "kritisk funktionalitet"
- ✓ Riskhantering
  - Mildra de mest kritiska riskerna
  - Följd av tester av arkitekturen.

Mjukvaruarkitektur

- ✓ Systemets nedbrytning
  - Hur delas systemet upp i delar?
  - Har vi alla delar?
  - Passar delarna ihop?
- ✓ Systemangelägenheter (concerns)
  - Systemegenskaper och kvalitéer
  - Avvägningar mellan systemkvalitéer
- ✓ **Systemintegritet**

Arkitektur är inte en utvecklingsfas!

- ✓ Den byggs upp av de mest centrala **designbesluten**.
- ✓ Av **nyckelabstraktioner** som kännetecknar mjukvaran
- ✓ Och **strukturen** av nyckelabstraktioner och centrala **interaktionsmekanismer**

Så…

- ✓ Under utvecklingsfasen kommer vi ställas inför ett stort antal strategiskt viktiga beslut.

> **Strategisk** — ***adj***
>
> ***Viktig eller nödvändig för en planering***
> ***Centralt för att upp nå önskade mål***

- ✓ Målet är ett system med förväntad
  - Funktionalitet
  - Kvalité
- ✓ **Vi behöver verktyg för att diskutera och resonera kring dessa beslut tidigt i processen.**

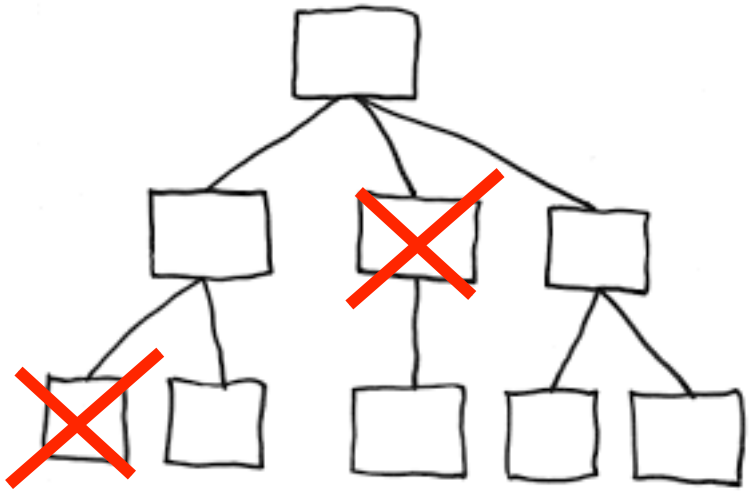
Arkitektur - Nedbrytning

- ✓ Ett system byggs upp av delsystem som i sin tur består av delsystem som i sin tur består av …
- ✓ Detta gäller alla system! Biologiska så väl som artificiella.

- ✓ Så problemlösning handlar om att
  1. Hitta de absolut bästa bitarna
  2. Foga samman dem

- ✓ Eller?

- ✓ Passar bitarna?



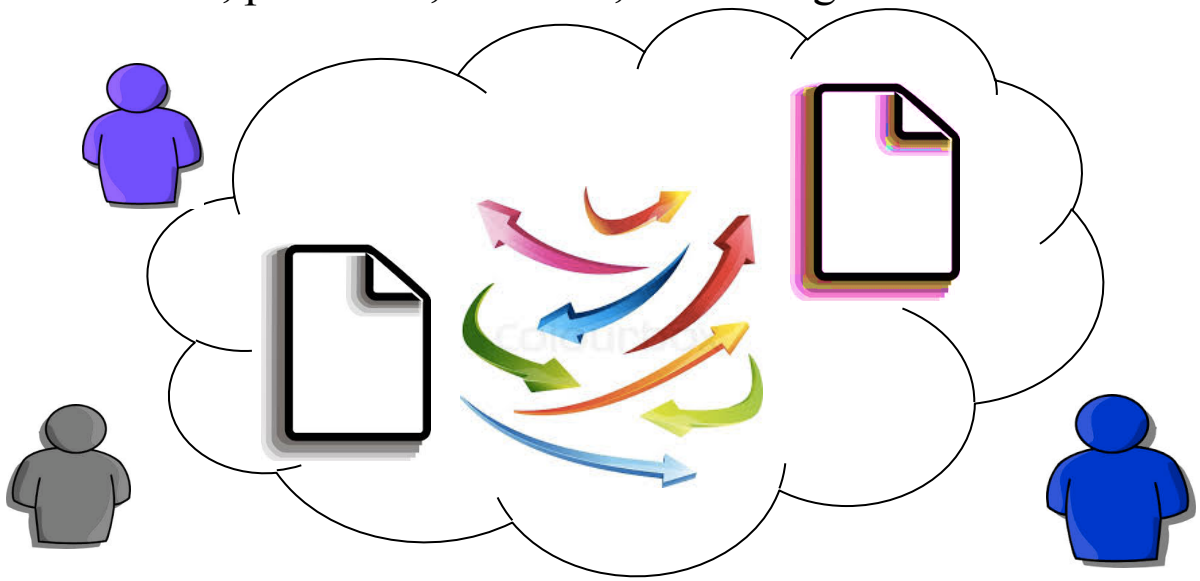
Systemintegritet

integritet - *(1)* - personlig integritet - skydd för privatlivet och privat information, rätten att ha hemligheter och att vara anonym; rätten att betraktas som hederlig och att slippa misstänkliggöras;
(2) - egenskap hos databaser, se referensintegritet;
(3) - systemintegritet (system integrity) - att datasystemet är komplett och oskadat - helt. - Ordet integritet, engelska integrity. beskriver egenskapen att vara hel, oskadad, odelad.

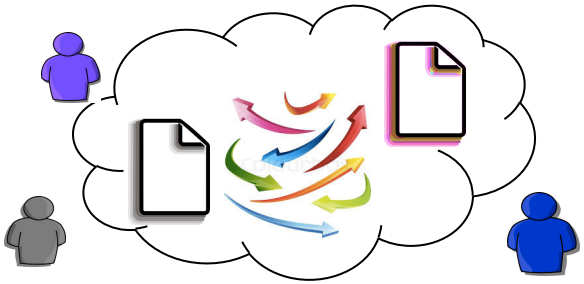
- ✓ De viktigaste systemegenskaperna först!!
- ✓ Vi kan inte uppnå en helhet, ett sammanhållet system, nedifrån och upp!
  - Göra nödvändiga avvägningar så att systemets egenskaper uppfylls när du bryter ned systemet.
  - Designa de arkitekturmekanismer som hanterar systemegenskapern, både interna och externa.

Ett exempel

- ✓ Fundera kring en applikationer där man kan “dela dokument med varandra”
- ✓ Vad är viktigast?
- ✓ Vilka krav har vi?
  - *Funktionella*, “dela filer”
  - *Kvalitet*, prestanda, säkerhet, tillförlitlighet



Hur dela upp i delar?



- ✓ Systemets nedbrytning
  - Hur delas systemet upp i delar?
  - Har vi alla delar?
  - Passar delarna ihop?
- ✓ System angelägenheter (concerns)
  - Systemegenskaper och kvalitéer
  - Avvägningar mellan systemkvalitéer
- ✓ Systemintegritet

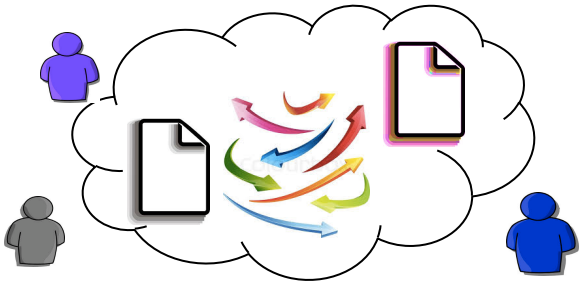
Funtionalitet

1. Identifiera delsystem och **ansvar**
2. Identifiera **gränssnitten** mellan delsystemen

Kvalitet

1. Identifiera **mekanismer**
2. Allokera ansvar till delsystem och gränssnitt.

Funktionalitet



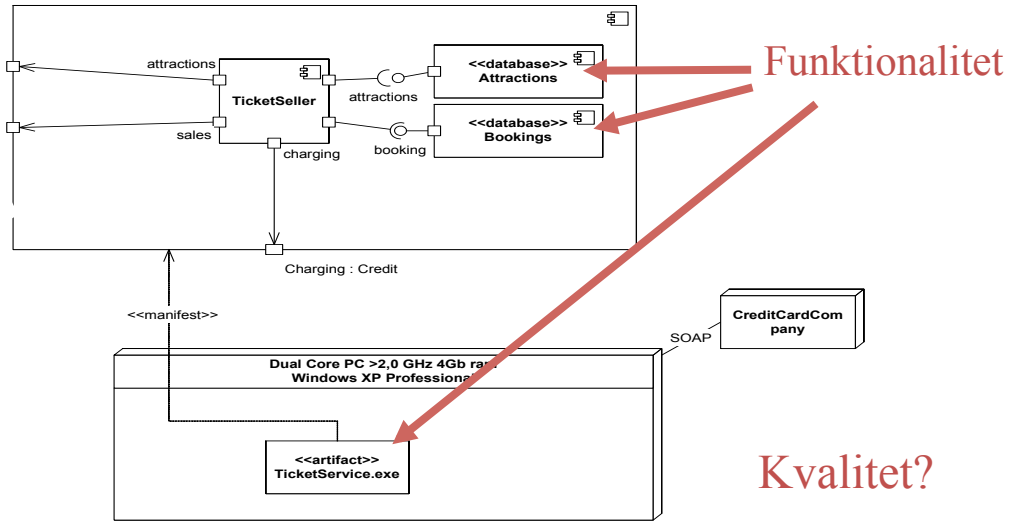
- ✓ Användningsfall – ger oss funktionalitet.
- ✓ Syfte:
  - Identifiera delsystem
  - Allokera tillräckligt med ansvar till delsystemen för att kunna uppfylla användningsfallen.
- ✓ Använd scenarier för att verifiera att du allokerat ”tillräckligt” med funktionalitet

Kvalitet

- ✓ “Cross-cutting”, systemets kvalitet finns i **hela** systemet, inte i enstaka delar.

- ✓ Systemkvalitet kan uppnås på två sätt i mjukvara
  1. Via vissa strukturegenskaper, mönster
  2. Via extrafunktionalitet i systemet, taktik

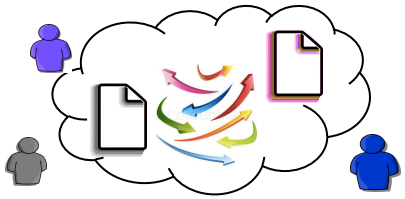
Integritet!!

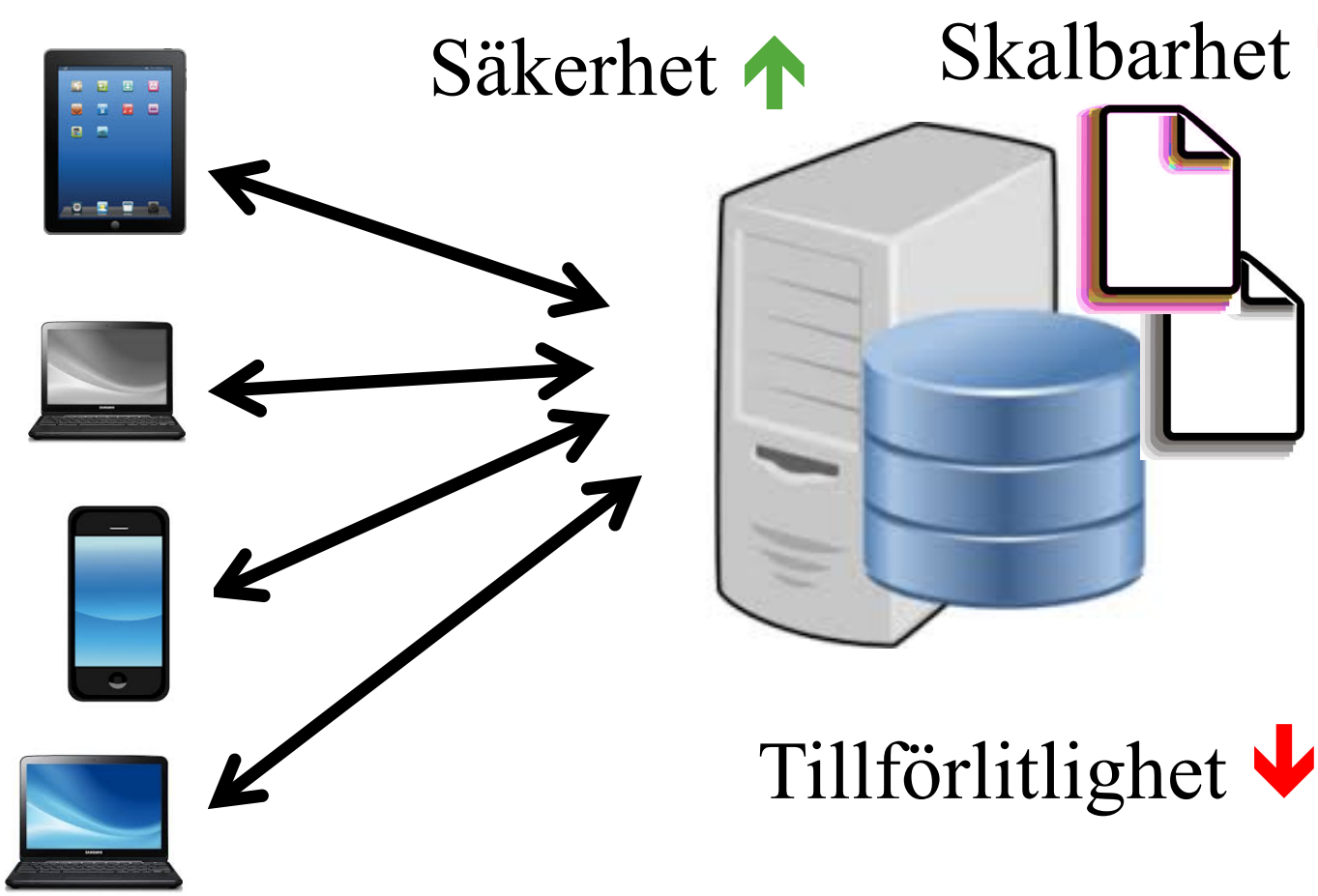


Funktionalitet

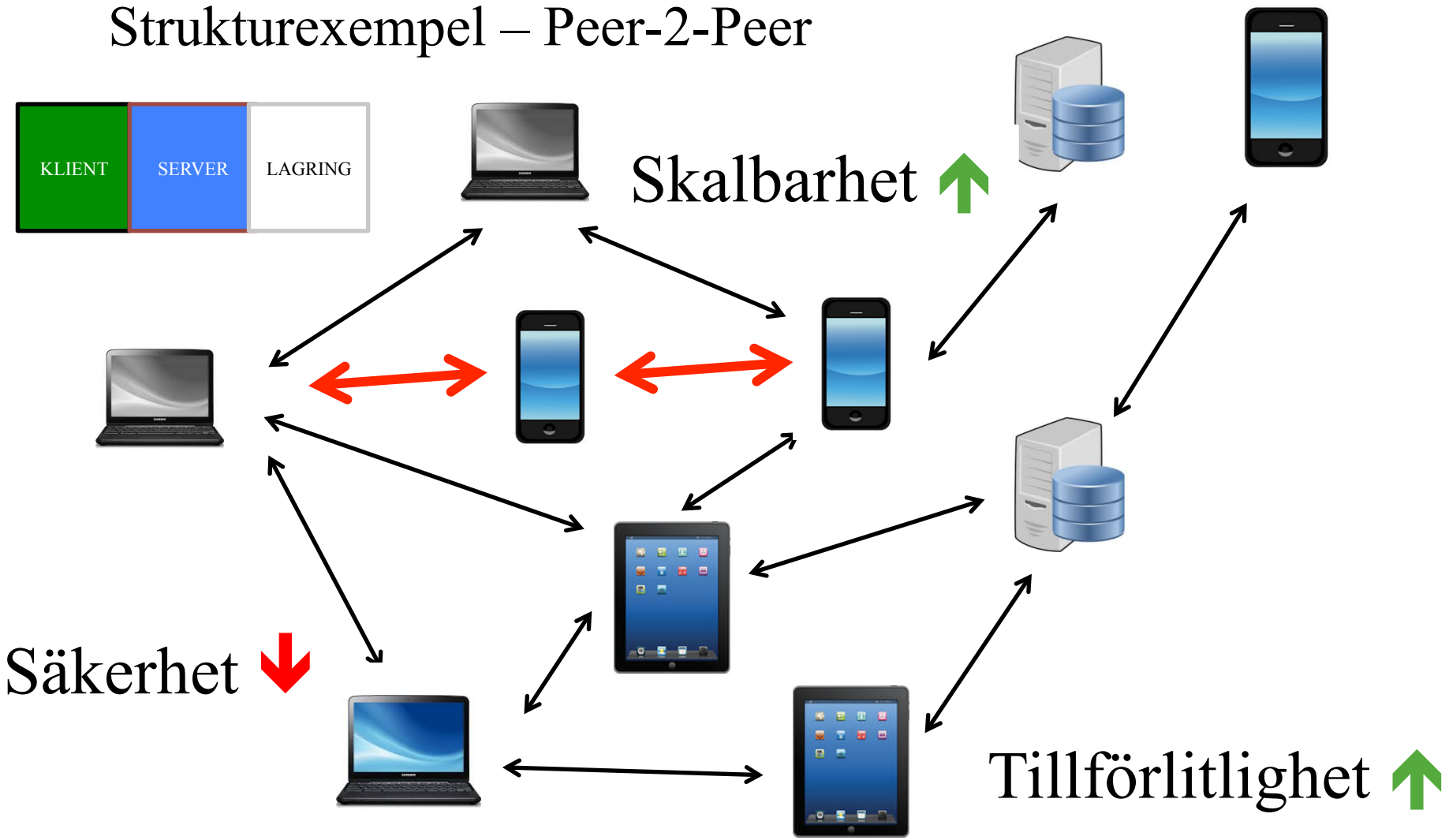
Kvalitet?

Strukturexempel – Klient-Server





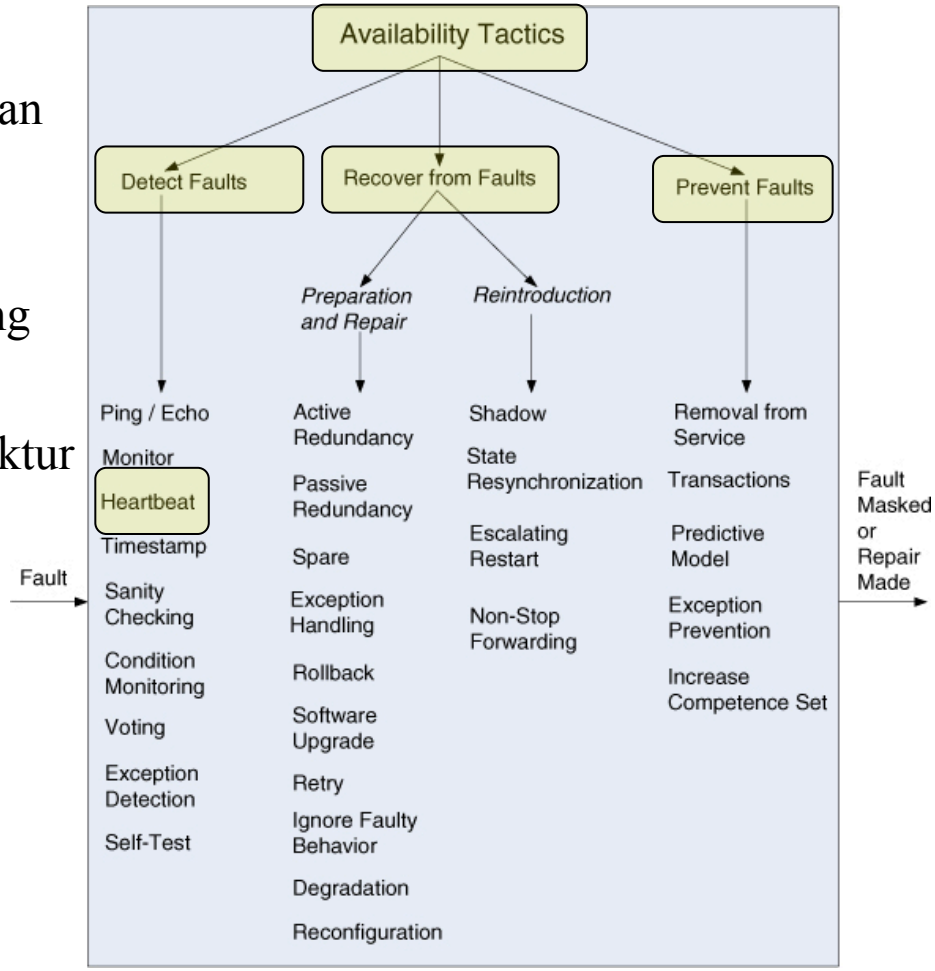
Strukturexempel – Peer-2-Peer



Taktiker

- ✓ Kvalitet som **funktionalitet** i mjukvaran
- ✓ Exempelvis
  - Prestanda genom lastbalansering
  - Tillförlitlighet med felåterhämtning

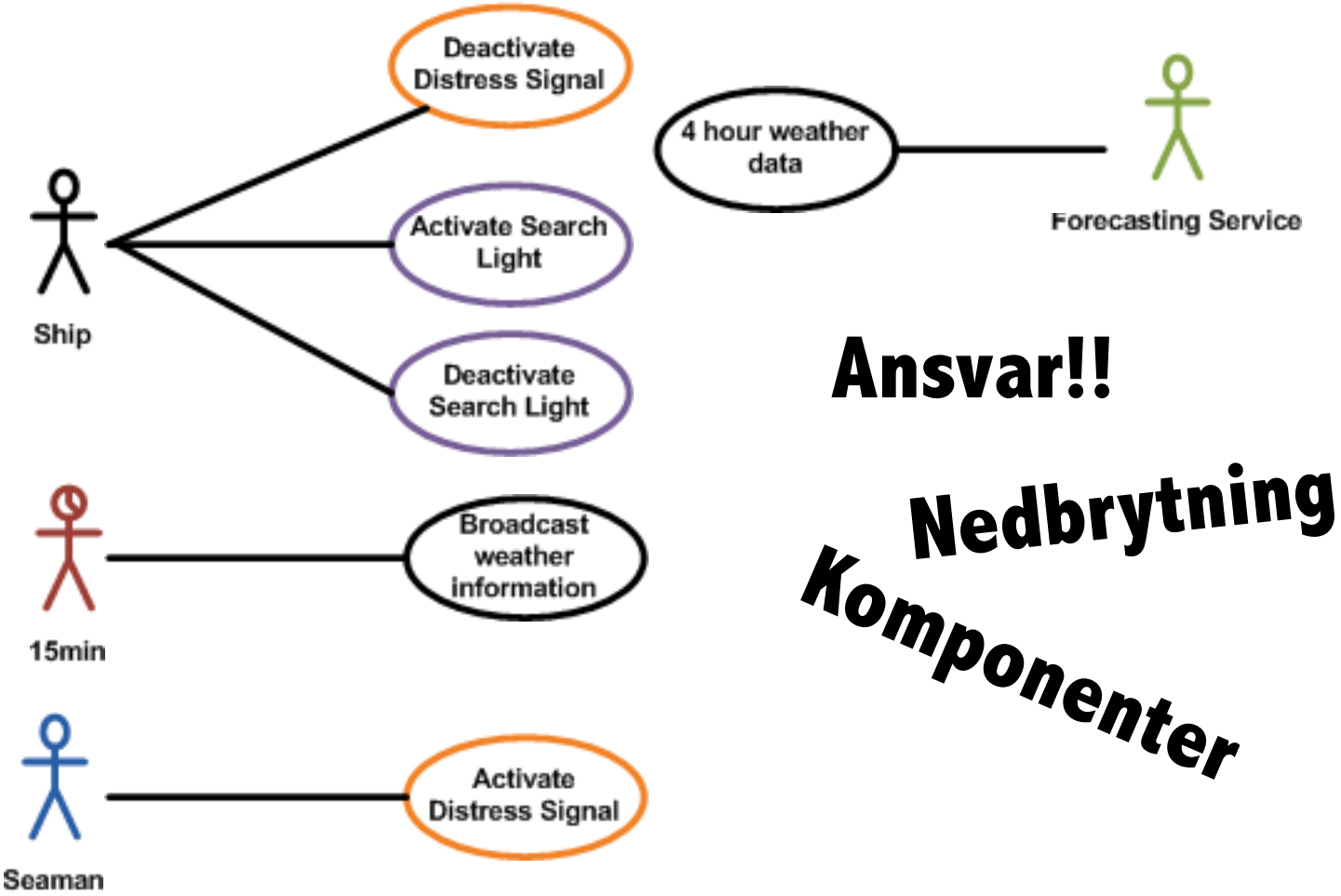
- ✓ Finns även taktiker som på verkar struktur
- ✓ Exempelvis
  - För att öka modifierbarheten
  - För att öka återanvändbarheten



Sea Buoy

In our seas there are hundreds of navigational buoys anchored near merchant fairways. Their main task is to provide the weather forecasting service and merchant navy with weather information, such as air and water temperature, and wind direction and speed. A buoy collects the weather information and broadcasts it every fifteen minutes via its radio transmitter. The buoy also has a radio receiver. The receiver is used by the weather forecasting services when it retrieves weather data for the last four hours weather and for other purposes such as maintenance by the company service personnel. The buoy can also be used as a lifeline for seamen in distress. If they reach a buoy and hang on to it, they can activate a SOS transmission and a flashing light by pulling an external switch. The buoy then transmits the SOS message until it is switched of via the radio receiver. In search operations vessels and liners can activate the buoy light. This is used for creating a search area grid.

Sea Buoy – Aktörer och Användningsfall



Ansvar!!

Nedbrytning

Komponenter

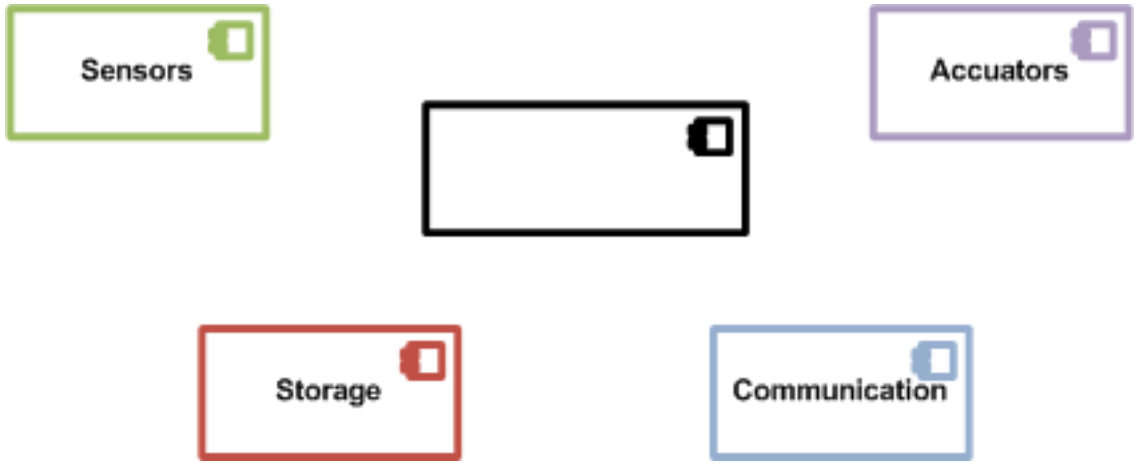
FURPS+

Sea Buoy

Supportability,

a. Uppgradering
b. Underhåll

Resurser?



Kommunikation?

Kvalitet och arkitektur

"En arkitektur möjliggör eller försvårar kvalitet"

Snarlika problem

Snarlika principer

Snarlika lösningar

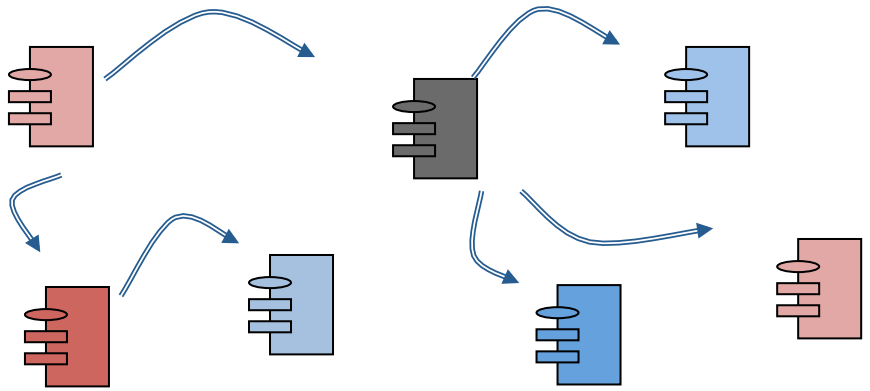
Organisationsnivån.

Organisation: (medeltidslatin organiza′tio, av organi′zo ’forma’, ’utgöra’, jämför organ), term inom organisationsteorin med två betydelser, dels en konkret där en planmässig samverkan mellan individer och grupper med gemensamma intressen åsyftas (förekommer ofta i sammansättningar, t.ex. personalorganisation), dels en mer allmän där ett företags eller en förvaltnings uppläggning av verksamheten avses.
De flesta organisationer har en uttalat hierarkisk utformning med klara skiljelinjer mellan över- och underordnade nivåer för att legitimera och underlätta beslutsfattande, ordergivning och kontroll.”

Det är ett system!

Implementationsnivån

- ✓ Imperativa programmeringsspråk (C, Ada, etc.)
  - Procedurer, funktioner, paket – erbjuder funktionalitet
  - Används av andra procedurer, funktioner, paket
- ✓ Objektorienterade programmeringsspråk
  - Objekt, skapar logiska enheter med data och funktion
  - Meddelanden, en abstraktion för användning av funktion

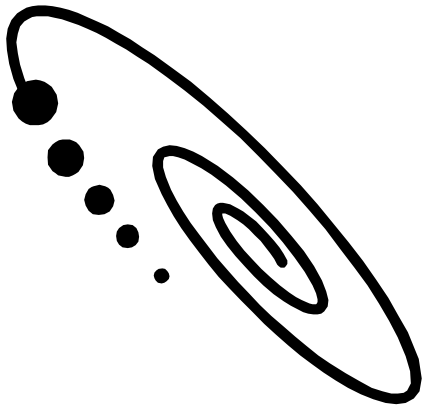


Det är ett system!

Arkitektur i UP

- ✓ Arkitekturen är central i iterativa och inkrementella processer

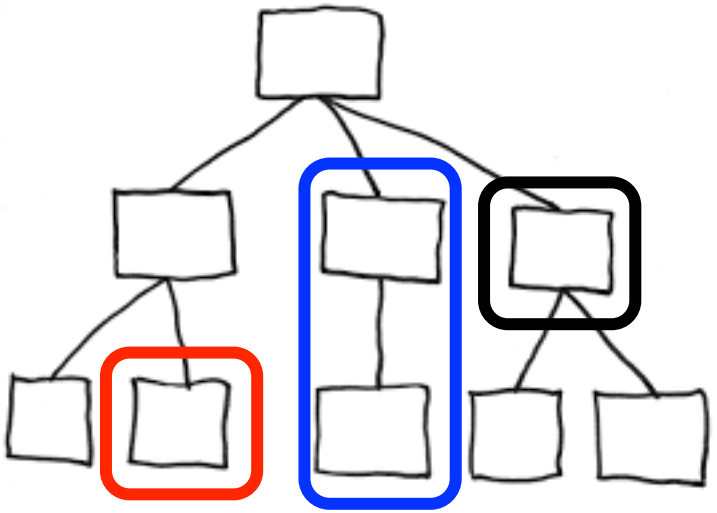
- ✓ I UP etablerar vi en arkitektur, tar fram en idé redan under inception-fasen
- ✓ Arkitekturen implementeras under elaboration-fasen
- ✓ Sedan förfinas den under konstruktionsfasen



ÅTERANVÄNDNING

Återanvändning

- ✓ Viktig del i elaboration-fasen
- ✓ Identifiera hur man skall återanvända komponenter
  - Bygga
  - Återanvända
  - Köpa



Typer av återanvändning

- ✓ Applikationer
  - Vi återanvänder och anpassar kompletta applikationer (COTS)
  - Utveckling av applikationsfamiljer (exempelvis. MS Office)
- ✓ Återanvändning av komponenter
  - Komponenter (t.ex. delsystem eller enstaka klasser) från en applikation återanvänds i en annan
- ✓ Funktionsåteranvändning
  - Små komponenter som implementerar en liten, väldefinierad, funktion.
  - Exempelvis metoder som sorterar heltal.

Värdet av återanvändning står i proportion till svårighetsgraden!

Opportunism vs. Systematik

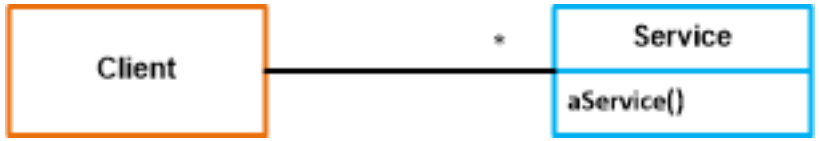
- ✓ Opportunisten, så fort det går och du vinner något så återanvänder du
  - Copy - paste
- ✓ Systematik, något helt annat…
- ✓ För att återanvändning skall bli riktigt effektivt måste det ske systematiskt!!

Systematic Reuse – the 3 hypoteser

- ✓ Vad måste vara uppfyllt för att det skall fungera?
  - *The Redevelopment Hypothesis:* Software development is actually Software Redevelopment. Most software development projects construct systems that are variations of existing software systems. It is more the rule than the exception that the new systems have more in common and the new parts when compared to existing systems.
  - *The Oracle Hypothesis:* It is possible to predict the types of changes that are likely to be needed to a system over its lifetime. In particular, the variation types on the abstract system architecture level.
  - *The Organizational Hypothesis:* It is possible to structure the software and its development organization in such a way that predicted changes and types are taken into account.

Exempel

Redevelopment



Oracle

Organizational

Återanvändning – Definition av två typer

- ✓ Återanvänd *Funktionalitet – ”Reuse will improve reliability and reduce project risks, time-to market will be shortened, and costs reduced”*
  - Ökad **tillförlitlighet**
  - Minskade **risker**
  - Kortare **tid-till-marknad**
  - Minskade **kostnader**
- ✓ Återanvänd *Design – ”Reused designs will shorten development time, provide “standard solutions” to common problems, and simplify maintenance”*

- ✓ Fler typer finns givetvis!

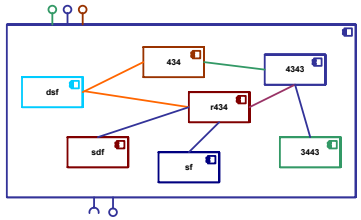
Exempel

- ✓ Code “scavenging” (copy & paste)
- ✓ Klassbibliotek
- ✓ Designmönster
- ✓ Distributionsramverk, t.ex. CORBA, EJB.
- ✓ Designåteranvänding (expertis)
- ✓ Objektorienterade ramverk
- ✓ Databashanterare (DBMS)
- ✓ Applikationsgeneratorer

Hur använder vi återanvändning

- ✓ Utveckling med återanvändning
- ✓ Utveckling för återanvändning
- ✓ Generator-baserad återanvändning
- ✓ Återanvändning av applikationer.

Utveckling med återanvändning



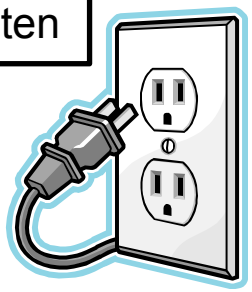
Arkitektur Design

Sök återanvändbara komponenter

Anpassning

Komponent Specifikation

Använd Komponenten



Utmaningar – Utveckling med Återanvändning

En ’algoritm’

- ✓ **Hitta rätt komponent**
  - – Svårt att matcha krav (sökmotor?)
  - – Lösning ➔ Skapa en Organisation för återanvändning
- ✓ **Använda komponenten**
  - – Anpassa komponenten
    - Interface adaptation (wrapping)
    - Invasive adaptation
  - – Integrera i systemet

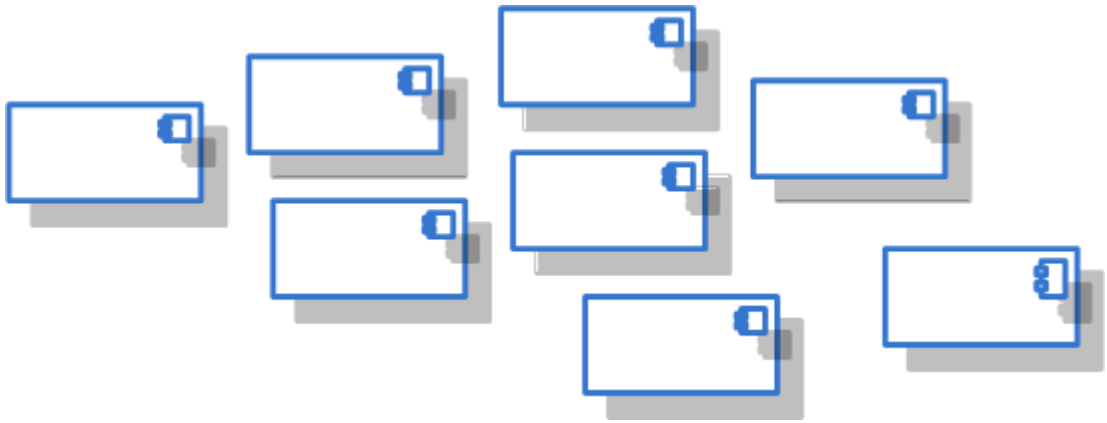
- ✓ Jämför *Opportunistisk* med *Systematisk*.

Problem med återanvändning

- ✓ Svårt att hitta **bra återanvändbara komponenter.**
- ✓ Komponenter kan vara svåra att förstå.
  - Metoder, ok!
  - Klasser, … också ok!
  - Subsystem, … inte lika enkelt.
- ✓ "**Not invented here**"

Utveckla för återanvändning

- ✓ Målet är att utveckla en uppsättning återanvändbara tillgångar.
- ✓ En större utmaning än traditionell utveckling.
  - Du vet inte hur tillgången kommer att användas och i vilket sammanhang.
  - Att skapa återanvändbara tillgångar är mycket svårt!



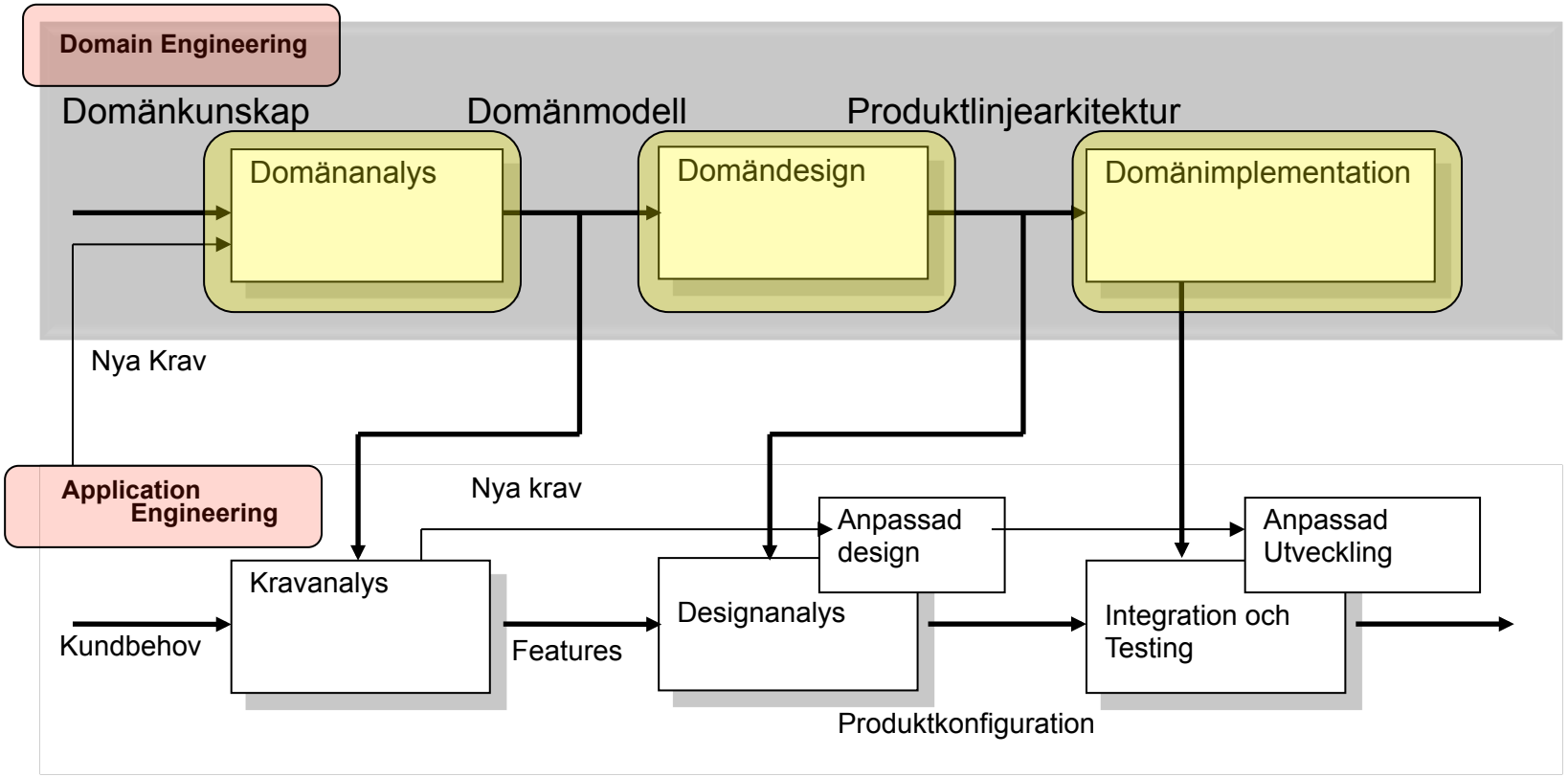
Utveckla för återanvändning – Hur

- ✓ Oracle
  - Vad skall finnas med
  - Vilken variation är tillgänglig.
- ✓ Organisation/Struktur
  - Mönster
  - Standardisera
    - Namn
    - Metoder, konstruktorer, operatorer mm.
  - Undantag (exceptions)
  - Generiska enheter (generics/templates)
  - Livscykelfrågor för återanvändbara tillgångar!

Produktlinjer för mjukvara

- ✓ A skapa en produktlinje är mer än att hitta användbara delar och skruva ihop dem lite ad-hoc.
- ✓ **Domain engineering** – metoder för att definiera familjer och ingående återanvändbara tillgångar.
- ✓ **Commonality and variability analysis** – process för att identifiera gemensamma och varierande delar inom en domän.
- ✓ Kombinerar *Utveckling för återanvändning* med *Utveckling med återanvändning*.

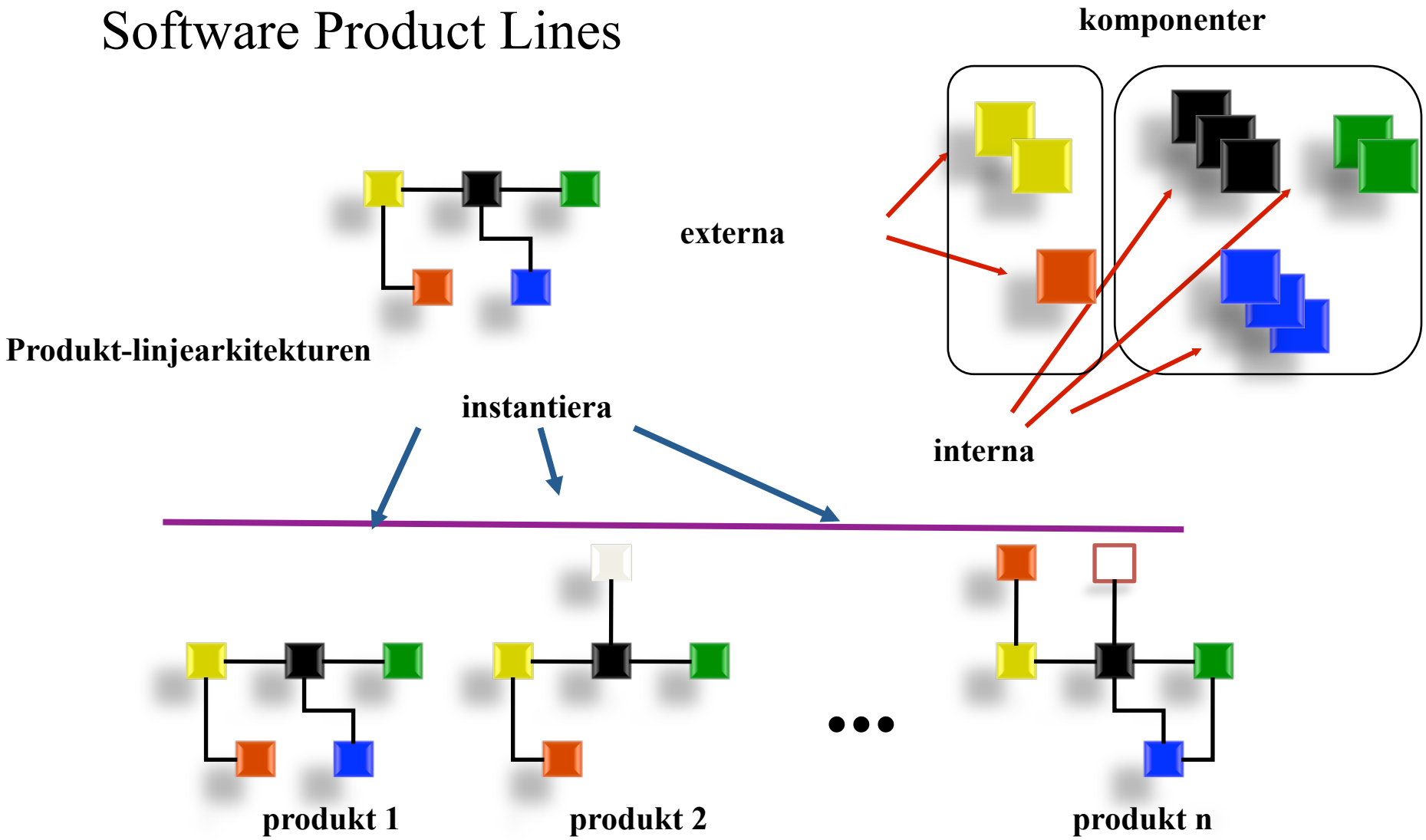
En processmodell med ’Domain Engineering’



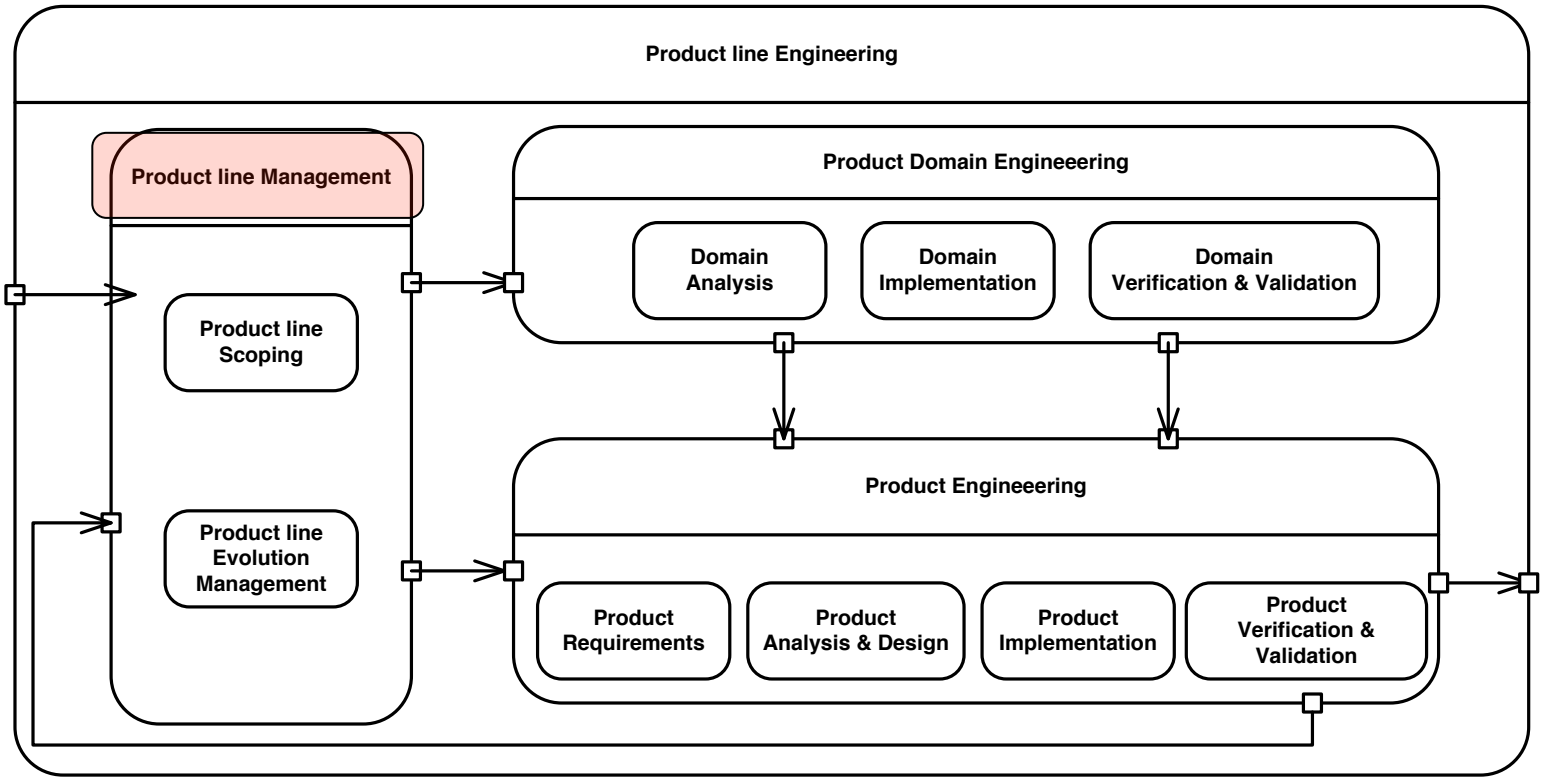
’Commonality’ och ’Variability’

- ✓ ’Commonality’-analys
  - Definiera gemensamma delar av produkter i en familj
  - **Feature,** en högnivåfunktionalitet (kundperspektiv) med given kvalitet.
- ✓ ’Variability’-analys
  - Identifiera alla produktspecifika ’features’
  - Union av dessa → produktfamiljens ’**variability’**

Software Product Lines



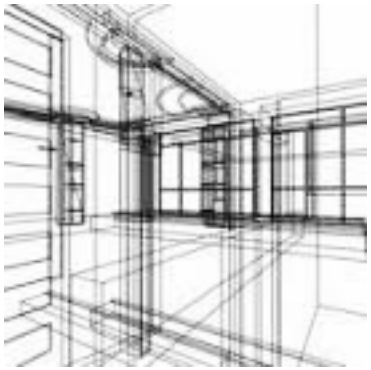
Software Product Line Engineering



Produktlinje ‘Scope’ – ‘Commonality’ and ‘Variability’

Återanvändning och Arkitektur

- ✓ Återanvändning är centralt för att snabbt bygga upp en basarkitektur
  - Ramverk
  - Plattformar
  - mm.
- ✓ Arkitekturen bryter ned systemet och beskriver delsystemens gränssnitt
- ✓ Utgör en bra grund för att leta efter lämpliga komponenter.



- ✓ Den verkliga arkitekturen i våra system bestäms av vilka komponenter vi (åter)använder.

I morgon

- ✓ Testning - Introduktion
- ✓ ”Software testing primer” på coursepress + kapitel 21 i boken.