Devoteam Group

Leader in Europe in IT Consulting

CONNECTING BUSINESS & TECHNOLOGY

DEVOTEAM QUAINT

DEVOTEAM QUAINT

Arkitektur på molnfri höjd – och neråt

Torbjörn Stavenek

torbjorn.stavenek@quaint.se

Intro

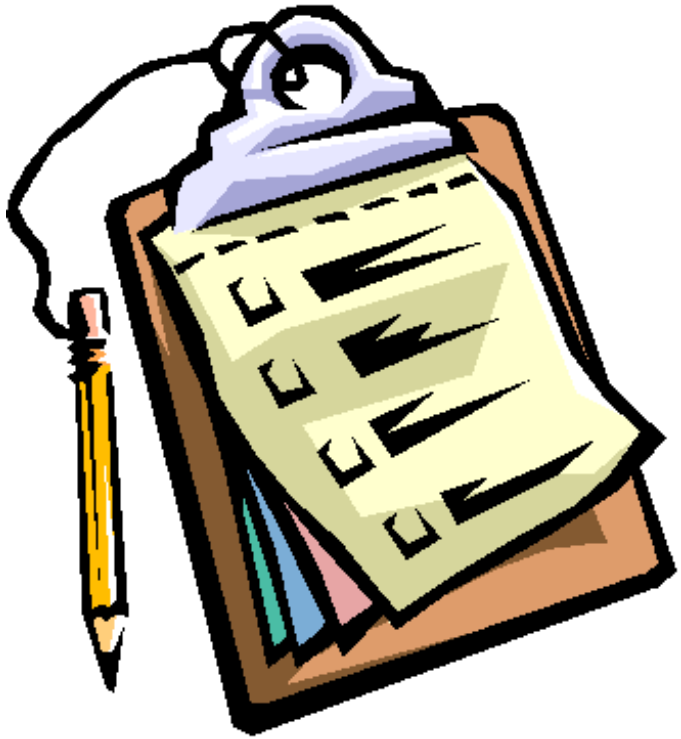
- Arkitektur handlar om avvägningar
- Vad ska prioriteras?
- Gemensamma nämnare

- Ledstjärnor

- Inspiration, lärdom och idéer
- Gränslandet mellan metod och arkitektur
- Ur arkitektens perspektiv

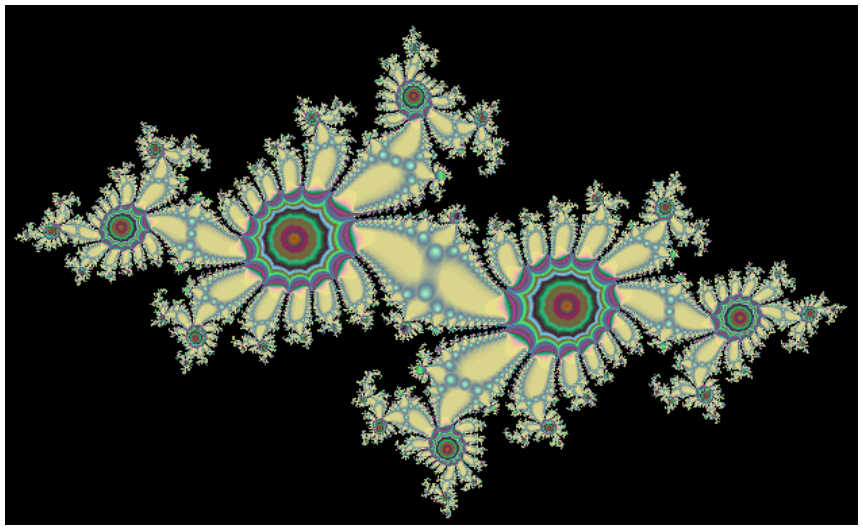
Agenda

- Jag
- Begrepp
- Ledstjärnor
- Applikationsarkitektur
- Arkitekturarbete
- Frågor



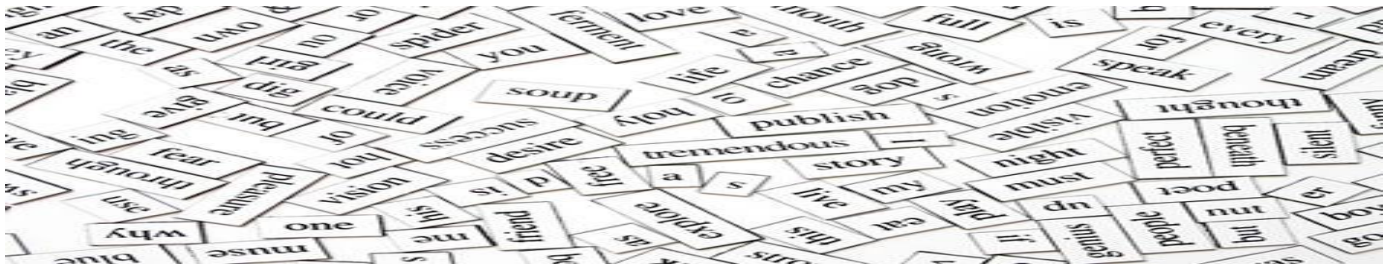
Jag

- 10 år i branschen
- Arkitekt, utvecklare, …
- Intressen
  - Verksamhet/IT
  - Arkitektur
  - Kvalitet
  - Konstruktionsmönster
  - Öppen källkod
- Erfarenheter
  - ICA
  - Rikspolisstyrelsen
  - Fora
  - SEB
  - Nordea



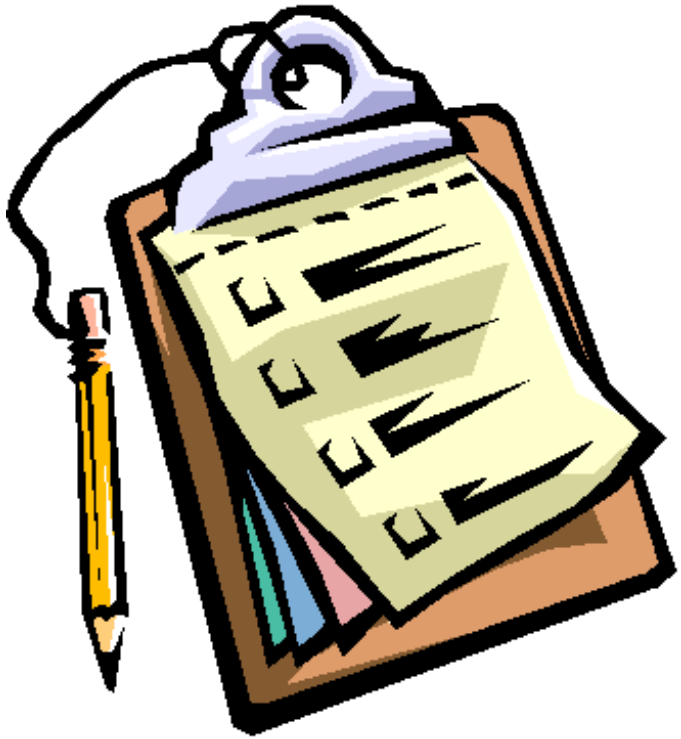
Begrepp

- BPM – Business Process Modeling
- DI - Dependency Injection
- EAR – Enterprise ARchive
- EDA – Event Driven Architecture
- ESB – Enterprise Service Bus
- JPA – Java Persistence API
- ORM – Object-Relational Mapping



Agenda

- Jag
- Begrepp
- **Ledstjärnor**
- Applikationsarkitektur
- Arkitekturarbete
- Frågor



Ledstjärna: Enkelhet

"Premature optimization is the
root of all evil."

- Donald Knuth

Ledstjärna: Enkelhet

- Lättförståelig - inte fattig/simpel

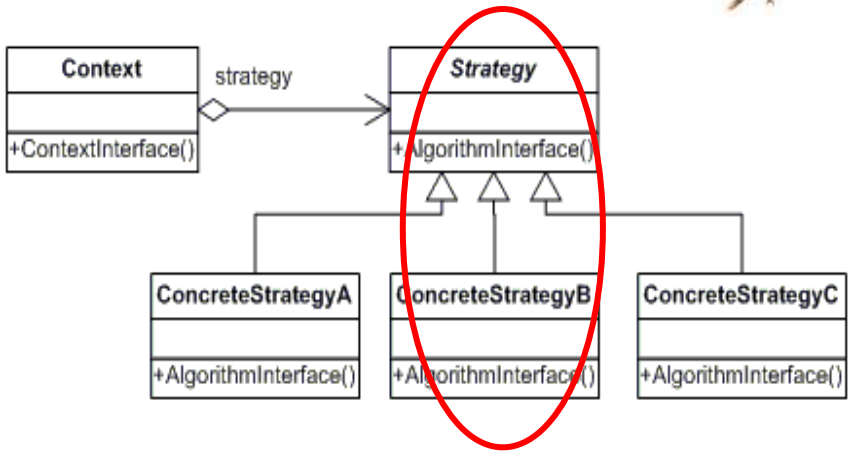
- Gör det enkelt för dig!
- Välj den enklaste algoritmen
- Använd ramverk som du och gruppen känner till
- Nytt ramverk - verkligen?
- Inkapsling av ramverk

- Bygg enklast möjliga system!

Ledstjärna: Minimalism

- Lite kod = lite arbete!
- Implementera bara kända krav!
- Slutliga visionen vs nästa levererans

- Undvik "bra o ha"-grejor
  - API:er
  - Abstraktioner
  - Strategimönstret
- Ramverk kan vara bra!



- Enkelhet vs Minimalism: Enkla lösningar vs lite kod
- “Minsta möjliga” och “Mest snabbimplementerat”

Ledstjärna: Återanvändning

- Lathet är en bra egenskap!
- Någon annan har redan gjort det du ska göra!
- Både öppen källkod och kommersiella produkter!
- Visst, det är kul att koda, men förvalta..?
- Har du full koll på Java?

Ledstjärna: Abstraktion

"Alla programmeringsproblem kan lösas med ett extra abstraktionslager..."

- David Wheeler

Ledstjärna: Abstraktion

- **Bra**
  - Skjut upp problem
  - Resonemang utan detaljer
  - Pseudokod oavsett arkitekturnivå
- **Dåligt**
  - Abstraktioner in absurdum!
  - Försvårar förståelsen
  - Prestanda

- **Rätt abstraktionsnivå för funktionalitet:**
  - En tjänst = en tydlig funktion/uppgift
  - Undvik sidoeffekter
  - Undvik teknik i namnet

Ledstjärna: Abstraktion

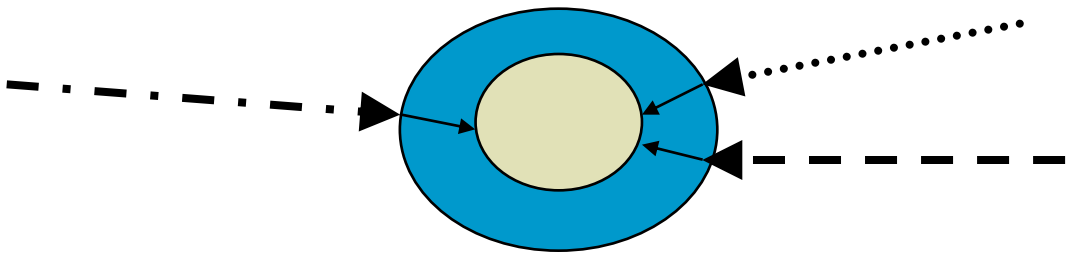
"Alla programmeringsproblem kan lösas med ett extra abstraktionslager...

...utom problemet med för många abstraktionslager."

- David Wheeler

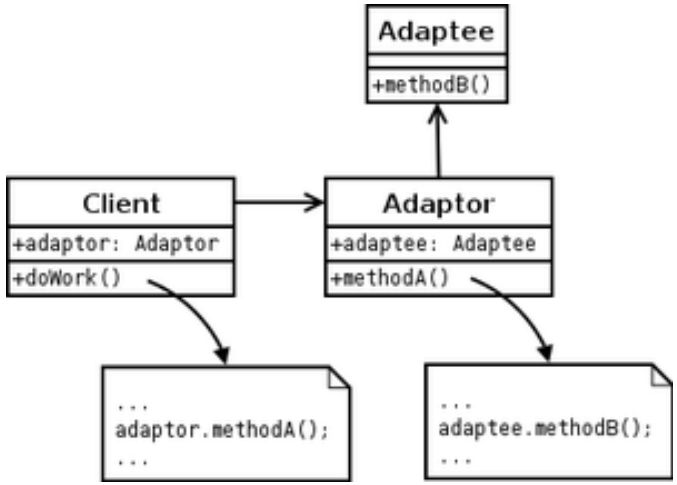
Ledstjärna: Konsekvens

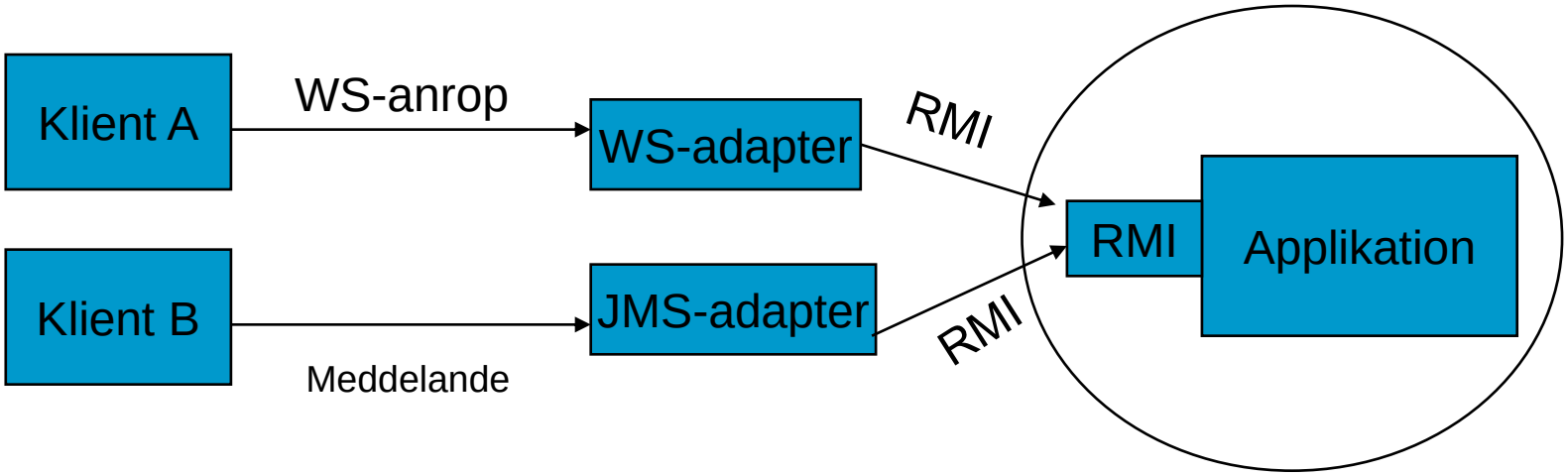
- Gör likadant hela tiden
  - API
  - Integration
  - Dataaccess
  - Protokoll
  - Namnstandarder
- Åtminstone i det *innersta* lagret runt applikationen



Konstruktionsmönster: Kommunikationsadapter

- Konstruktionsmönster
- Adaptermönstret på arkitekturnivå
- Inre konsekvens
- Nya integrationer enkelt





Ledstjärna: Fokus

- Vad är systemets ansvar?
- En av arkitektens viktigaste uppgifter
- Arkitekt = Polis
- Ifrågasätt systemets delar
- Ett system - en uppgift!
- Systemets syfte vs ny funktionalitet?

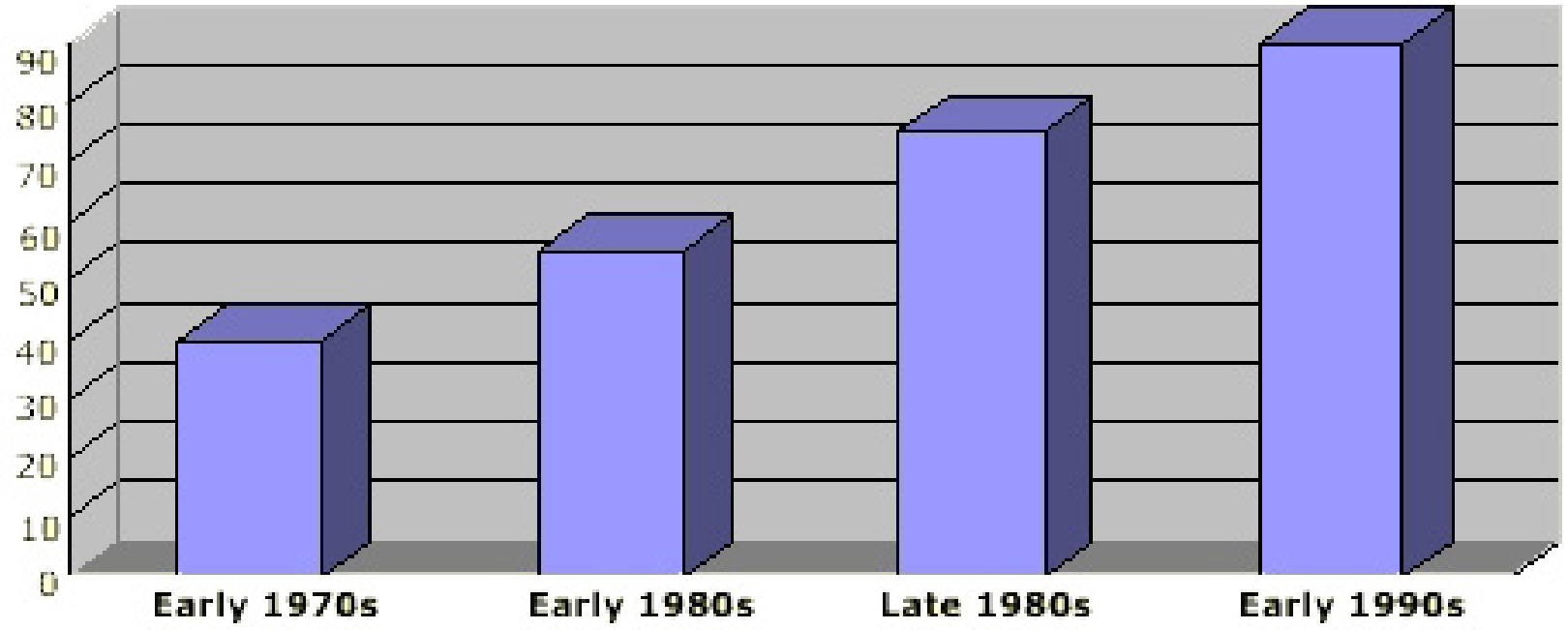
Ledstjärna: Formalism

- Var formell när det hjälper
  - För att undvika frågetecken
  - För att minnet sviker…
- Ifrågasätt dokumentens syfte!
- Vem är mottagaren?
- Dokumentation – på en rimlig nivå!

Ledstjärna: Holism - helhetssyn

- Arkitekten ska ha koll på visionen
- Visionen guidar detaljbesluten
- Om vi väljer X istället för Y:
  - Hur påverkas…?
  - Frågetecken?
  - Prototypas?
- Modifiera kravbilden och få ett "bättre" system?
- Se bortom nästa ”krök”
- Följer helheten ledstjärnorna?

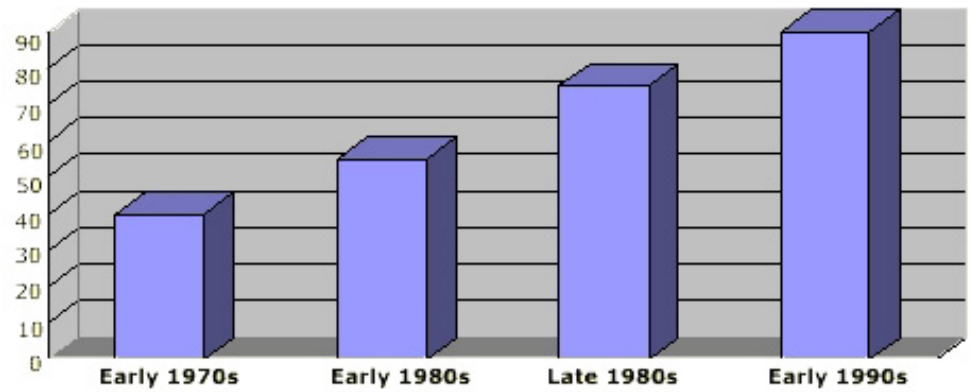
Ledstjärna: Minimera förvaltning



Procent av livscykelkostnad för mjukvara för underhåll och förvaltning (Moad 1990)

Ledstjärna: Minimera förvaltning

- Förutse vad som kommer att ändras
- Följ standarder
- Ledstjärnorna
- Dokumentation och dess relation till personalomsättning
- 95% förvaltning – inhouse eller..?
- Några dyra projekt…

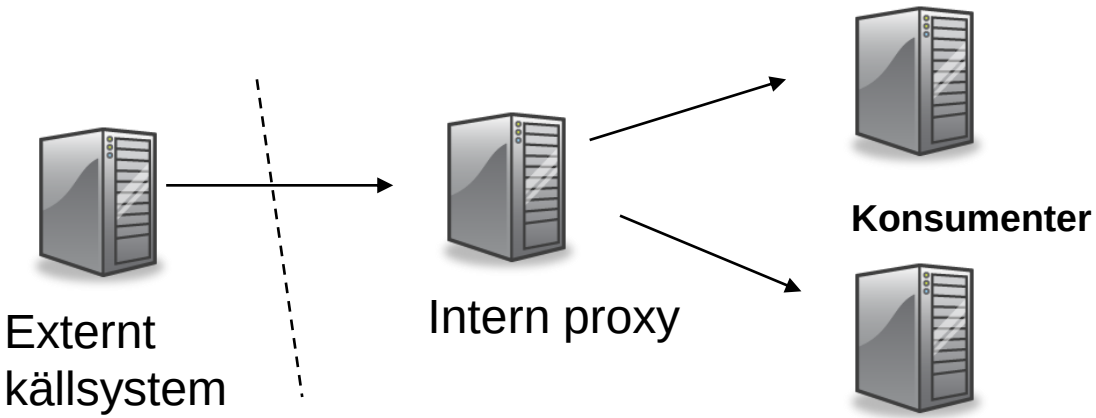


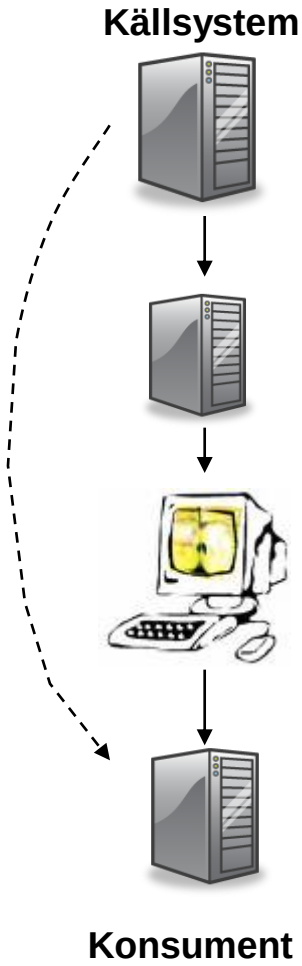
Procent av livscykelkostnad för mjukvara för underhåll och förvaltning (Moad 1990)

Ledstjärna: Undvik transitiva beroenden

- Definition: hämta data via mellanhänder
- Hämta alltid data från källan
- Förändringar i källsystemet – många ändringar
- Alla system i kedjan måste vara uppe
- Fördröjningar

- Undvik i ditt eget systemlandskap
  - Undantag: proxies mot externa system

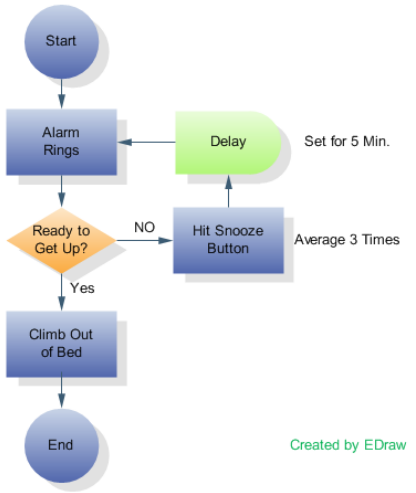




Ledstjärna: Deklarativt vs imperativt

- Koda med "vad" istället för "hur"
  - Annotations (speciellt JPA)
  - CMT – Container Managed Transactions
  - Regelmotorer
  - BPM – Business Process Modeling
- Högre abstraktionsnivå!
- Färre rader kod!
- Kompilator i Prolog, någon?

- Men - vi måste känna till detaljerna!



Ledstjärna: Standarder

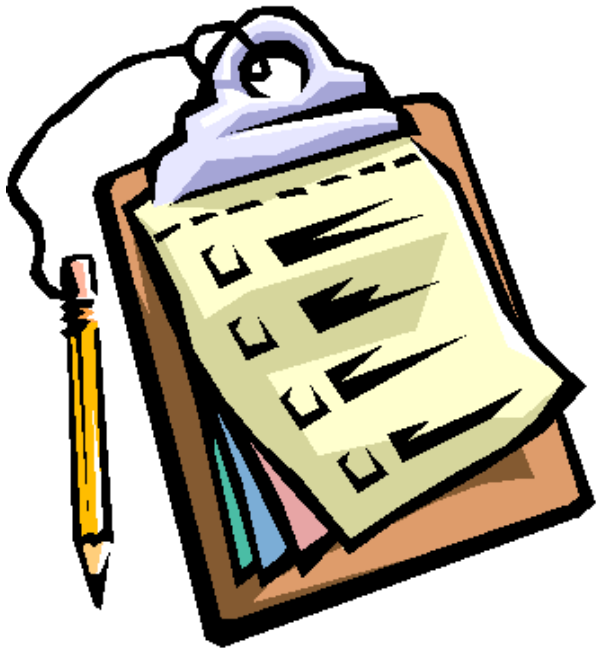
"Standarder är bra, alla borde ha en!"

Ledstjärna: Standarder

- Minskar dokumentationsbördan
- Kunskap mellan projekt
- Nyinlärning bygger generell kompetens
- Olika typer
  - de jure – "av rätt", officiell (Entitetsbönor i EJB2)
  - de facto – i praktiken (Hibernate)
- Båda är viktiga!
  - de jure – vid krav på olika målplattformar
  - de facto – viktig kunskap mellan projekt

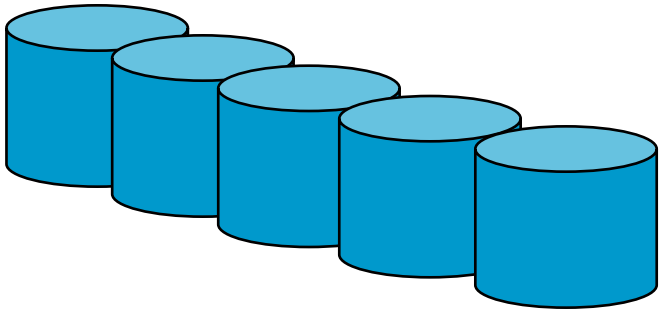
Agenda

- Jag
- Begrepp
- Ledstjärnor
- **Applikationsarkitektur**
- Arkitekturarbete
- Frågor



Applikationsarkitektur: Undvik dubbellagring

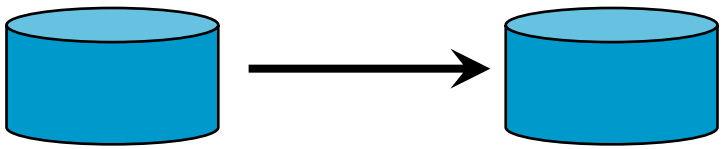
- Vad är det för data?
- Behandla direkt?
- Jobba med metadatan?
- Infrastrukturen felaktig?
- Dubbellagring
  - kostsamt (fysisk disk, tid för backup:er, förvaltning, …)
  - Kan ge felaktiga data (synkroniseringsproblem)
- Samma datastruktur?! Oftast opassande…



Applikationsarkitektur: Undvik direkta databasladdningar

- DB-laddningar kringgår all implementerad verksamhetslogik
- Kan skapa
  - Inkonsistenser
  - Samband som inte "finns“ eller skulle godkänts
- Undantag pga prestanda

- Men om vi flyttar ner logiken till databasen då..?



Applikationsarkitektur: Undvik logik i databasen

- Direkt DB-laddning… Lägg logiken där?
- Verksamhetslogik i sql?!
- …eller dubblerad logik?!
- Smidigt att förvalta?
- Välfungerande driftsättningscykel?
  - Diff-skript?
  - Bortglömda ändringar
- Undvik SQL - ersätts av ORM:ar!

Applikationsarkitektur: Konfiguration

”Hårdkoda mera!”

Applikationsarkitektur: Konfiguration

- Koppla ihop – hur?!
- Hårdkodning
  - Enkelt
  - Snabbt
  - Förståeligt
  - Men ohållbart…
- Egen Factory + properties-fil
  - Proprietärt…
- XML-DI, tex Spring
  - Run-time check…
- Annotations-DI,
  tex Ejb3, Spring 2.1+, Google Guice
  - Compile-time check!

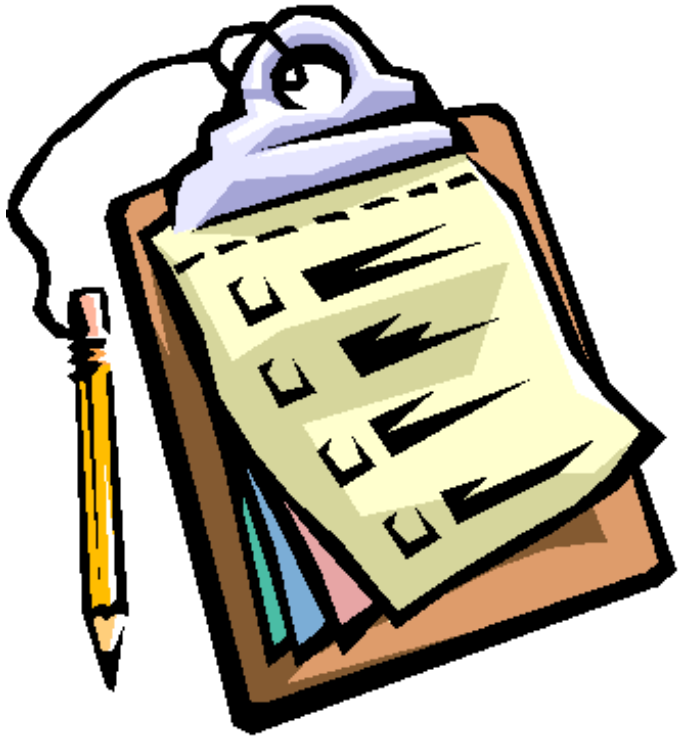
Applikationsarkitektur: Implicit konfiguration

- Configuration by convention
- Configuration by exception
- Bra exempel
  - Hibernate
  - Maven
  - JPA med annoteringar
- Skönsvärden och "Att bara starta!"
  - Konfig-fil + dok, eller bara en konfig-fil?
  - Configuration by exception
- Varför ha påtvingad konfiguration?

- Men - vi måste känna till detaljerna!

Agenda

- Jag
- Begrepp
- Ledstjärnor
- Applikationsarkitektur
- **Arkitekturarbete**
- Frågor



Skapa en ny arkitektur

- Fundera igenom alla ledstjärnor
- Vad ska gälla för detta systemet?
- Vilken är den stora generella riktningen vi ska ha?
- Vilka ledstjärnor är viktigast?

- Time-to-market?
- Stabilitet?
- Produkt?
- Internt ramverk?

Göra om en befintlig arkitektur

- Var tjänar vi mest? (Minimalism, Konsekvens, Holism)
- Mycket buggar? (Enkelhet, Minimera förvaltning)
- Hur ser skiktningen ut? (Enkelhet)
- Ramverk? (Enkelhet, Minmalism, Standarder)
- Minska kodmängden? (Minimalism, Minimera förvaltning)
- "Tabula rasa" – tomt blad? (Enkelhet, Fokus, Holism)
- Gå mot SOA eller EDA (Fokus, Holism, Konsekvens)

Ledstjärnor – ”…och vinnarna är…”:

- Enkelhet
- Konsekvens
- Minimera förvaltning

Frågor

Slut

Testbarhet

- Automatiserbara tester = bra!
- Hur skapar man en testbar arkitektur?
  - **Lös koppling**
  - **Interface-mönstret**
  - **Depencency Injection**
  - **Factory-klasser**

- OBS!
  - **Testkod kan göra vad som helst!**
  - **Produktionskod får *aldrig* påverkas av hur den testas!**

Diskutera

- Diskutera med kollegor
  - **Olika uppfattningar om bra arkitektur**
  - **Olika uppfattningar om vad som är viktigt**
  - **Olika uppfattningar om problemen i senaste projektet**

- Brainstormingens principer
- Underskatta inte social interaktion!

Projektgrupp

- Paragrafryttare/metodfascister
- Värdet av ”Bra folk”™

Pragmatism

- Var praktisk och verklighetsförankrad
- Problem! Hur kan vi lösa det?
- Välj en fungerande lösning och gå vidare!
  - Fastna inte på att din planerade lösning inte fungerade
  - Lös problemet och gå vidare
  - Lös de stora problemen först
  - Återkom senare…
- Tex du ska lösa X mha ramverk Y, men problem…
  - -”Inte som planerat…”
  - -”Inte perfekt…”
  - Funkar det? Ja, då så!

- Paragrafryttare – 10 sätt att känna igen dem… nej.
  Var inte en, ha inte med dem i projekten…

Löst kopplat

- Denna har ni hört förr eller?!
- Arkitekturnivå
  - ESB och EDA – löst kopplade tjänster och system
  - Kommunikationsadapter – löst kopplad applikation
  - Pipes and filters – kedjor av frikopplad logik
- Klassnivå
  - Dependency Injection
- Prestanda

- Men glöm inte: enkelhet! (Har du för mycket "träd"?!)

Ifrågasätt

- Varför ska detta vara med?
- Hur blir systemet bättre om vi gör si eller så?
- Varför har man gjort så?
- Varför finns detta kravet?

Kommunikationsstrategier

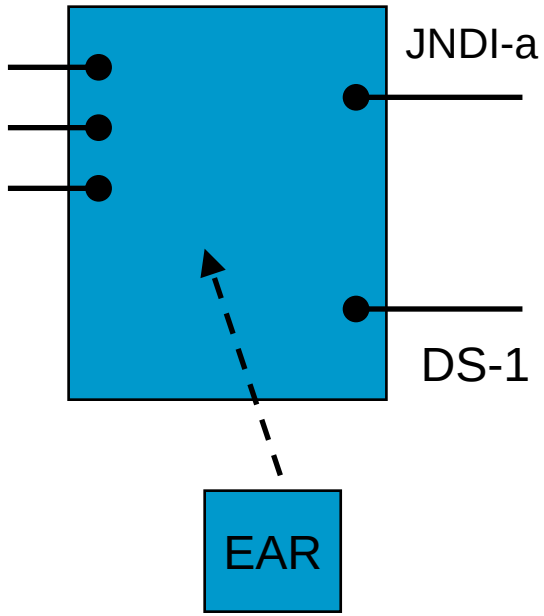
- Lite buzzwords
- Ta emot data vid *publicering av händelser* (EDA)
  - **Trevligt**
  - **Uppdateringar "automatiskt"**
  - **Kontinuerligt litet dataflöde**
  - **Bra utnyttjande av infrastrukturen**

- Hämta från publika tjänster *on-demand (SOA)*
  - **Ger typiskt stora datamängder**
  - **Ger enstaka stora "spikar" vid överföringar**
  - **Kan kräva mycket CPU/bandbredd som sällan utnyttjas**

- EDA + SOA = Sant!

Extern konfigurationsfil

- Dvs bygg inte in konfigurationsfiler i EAR:en
- Målmiljön erbjuder tjänster
- Dock två typer av konfigurationer
  - Intern – kan byggas in
    - Saker som sällan ändras
    - Tex konstanter för JNDI-namn
  - extern – undvik att bygga in
    - Saker som ändras mellan miljöer
    - Vid testning
    - När omvärlden ändras
    - Tex datakällor
- Vad vinner man på detta då?



Sträva efter en enda EAR

- En enda EAR byggs, testas och driftsätts

- Fördelar
  - Enklare byggen
  - Färre profiler
  - Snabbare lokal testning av byggen
  - Ny konfiguration behöver bara byggas vid ändring av målmiljö eller tillägg av nya
- Nackdelar
  - Målmiljöer måste konfigureras separat
  - Konfigurationer kan kräva egna byggen

Contact

AUSTRIA

BELGIUM

CZECH REPUBLIC

DENMARK

FRANCE

MOROCCO

NETHERLANDS

NORWAY

POLAND

SAUDI ARABIA

SPAIN

SWEDEN

SWITZERLAND

UNITED ARAB EMIRATES

UNITED KINGDOM

CONTACT

Person: Torbjörn Stavenek

Phone: 0733-812107

E-mail: torbjorn.stavenek@quaint.se

Address

DOCUMENT

ID: #

Author:

Date: