



ADSE2200 - Systemutvikling Arkitektur

Høgskolen i Oslo og Akershus
Avdeling for ingeniørutdanning
29. september 2014

Suhas G. Joshi – joshi@ifi.uio.no

Arkitektur

"Perfection (in design) is achieved not when there is nothing more to add, but rather when there is nothing more to take away."

(Antoine de Saint-Exupery)

"We try to solve the problem by rushing through the design process so that enough time is left at the end of the project to uncover the errors that were made because we rushed through the design process."

(Glenford Myers)

"Architect: Someone who knows the difference between that which could be done and that which should be done."

(Larry McVoy)

Oversikt

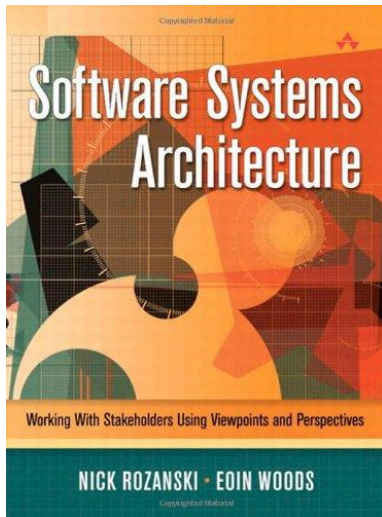
- Arkitektur
 - Hva er arkitektur og hvorfor bry seg?
 - Hvem er arkitekten og hvordan jobber hun?
- Arkitektoniske modeller
- utfordringer ved arkitektur
 - Designnivåer
 - Hvem og hva påvirker en arkitekt?
 - God systemarkitektur
 - Arkitektoniske drivere og systemdesign
 - Forhold mellom ulike designmål tilknyttet arkitektur
- Arkitektoniske stiler
 - Ulike typer arkitektur
 - (Devilry, Netflix og) Facebook sin arkitektur
- Service-oriented architecture (SOA)

Læringsmål

På alle **røde lysark** står det en forklaring på nøyaktig hva det forventes at dere skal kunne svare på.

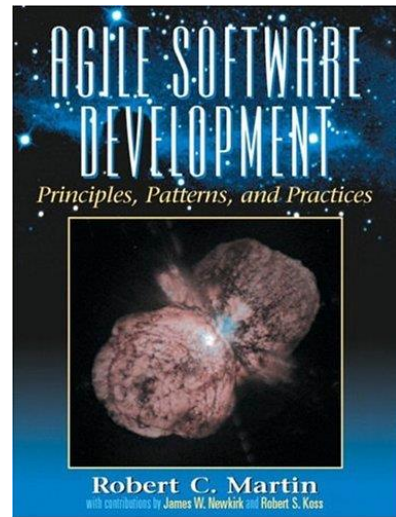
- Fargeforklaring på lysark
 - **Rød** – MÅ læres. Relevant for ukesoppgaver, obliger og eksamen. *Kjernepensum.*
 - **Oransje** – BØR læres. Nyttig å kunne trekke inn i oppgaver til eksamen. *Statarisk pensum.*
 - **Grønn** – KAN læres. Ikke pensum, men fint for å imponere gruppelærer/sensor. *Kursorisk pensum.*
- Temaer som må kunnes etter i dag:
 - Hvem er arkitekten som interessant, og hva er arkitektens rolle i systemutviklingsprosjektet?
 - Hvorfor er arkitektur vanskelig? Hvordan kan man måle god arkitektur?
 - Hvilke ulike typer arkitekturstiler finnes, og hvilke fordeler/ulempes har disse?
 - Hvordan skiller man mellom fysisk og logisk arkitektur?
 - Hva er tjenesteorientert arkitektur?
 - + øvrig pensum i kapittel 7 (?)

Videre lesing for de interesserte...



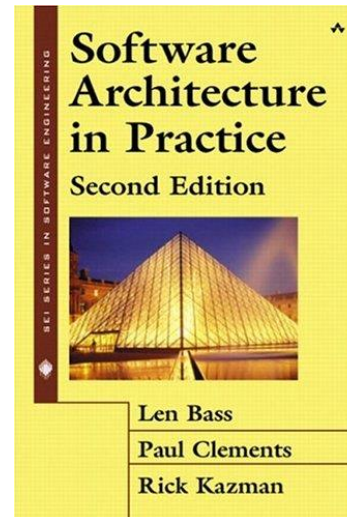
Software Systems
Architecture: Working With
Stakeholders Using
Viewpoints and Perspectives

*Nick Rozanski, Eóin Woods
(2005)*



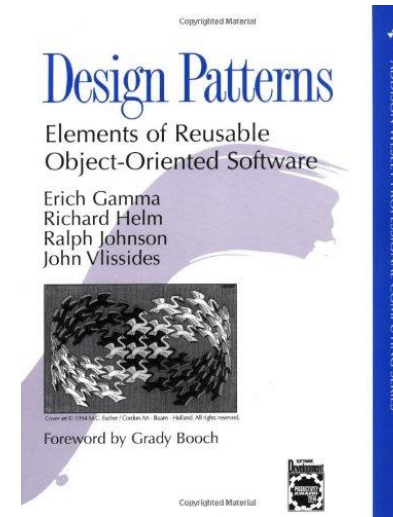
Agile Software
Development, Principles,
Patterns, and Practices

Robert C. Martin (2002)



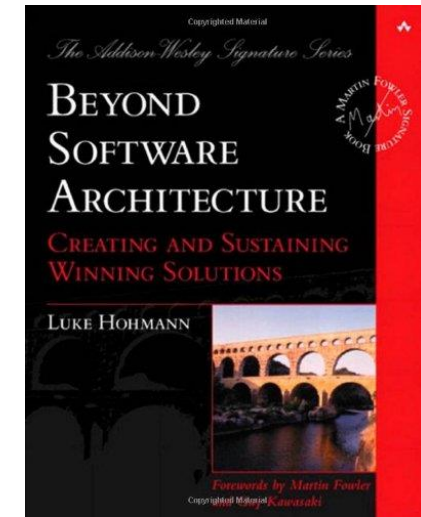
Software Architecture in
Practice

*Len Bass, Paul Clements,
Rick Kazman (2003)*



Design Patterns: Elements
of Reusable Object-
Oriented Software

*Erich Gamma, Richard
Helm, Ralph Johnson, John
Vlissides (1994)*



Beyond Software
Architecture: Creating and
Sustaining Winning
Solutions

Luke Hohmann (2003)

Arkitektur

- Alle bygninger har en bygningsarkitekt, og alle systemer har en systemarkitekt
- En bygningsarkitekt kan tegne et bygg elegant og effektivt eller klumsete og dysfunksjonelt
- ... og det kan også systemarkitekten!
- Husk:
 - Arkitektur er en eldgammel kunst
 - System- og programvareutvikling begynte i etterkrigstiden
 - Systemarkitektur er en mye mer moderne disiplin

Arkitektur

In the Media

2 comments Listen Pri

IBM new computing architecture based on human brain



By Eduardo Arrufat
Aug 8, 2013 in [Science](#)

LIKE THIS ARTICLE

17

SHARE



0



Share

42



Tweet

8



Share

0



GOOGLE +

0

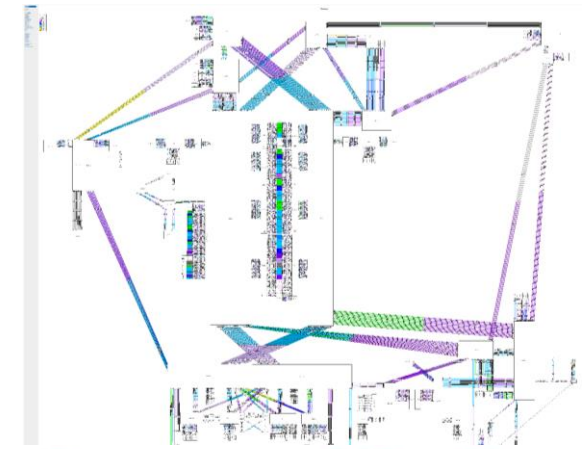


[London](#) - IBM has unveiled a new computing architecture on Wednesday whose inspiration comes from the human brain. Scientists hope that it will achieve brain level capabilities.

[Many attempts](#) have been done in the past that try to measure or evaluate the capacity and potential of the human brain. We have landed a man on the Moon and we have been to the deepest of the oceanic trenches;

however, the human brain is still a mystery to us.

Nevertheless, a new trend in the scientific community might be gestating over at the [Watson Research Center](#). Instead of trying to discover how the brain works by observation, why not try to replicate its architecture and then analyze the behavior. That is what IBM scientist showed on Wednesday, an [all-new computer architecture](#) that is based around the complexity of the human brain.



FACEBOOK DRONES INTERNET

Facebook wants to use drones to blanket remote regions with Internet

Robin Burks Wednesday, March 5, 2014 - 12:34pm



Arkitektur

How The AOL.com Architecture Evolved To 99.999% Availability, 8 Million Visitors Per Day, And 200,000 Requests Per Second

MONDAY, FEBRUARY 17, 2014 AT 8:56AM

This is a guest post by [Dave Hagler](#) Systems Architect at AOL.

The AOL homepages receive more than **8 million visitors per day**. That's more daily viewers than Good Morning America or the

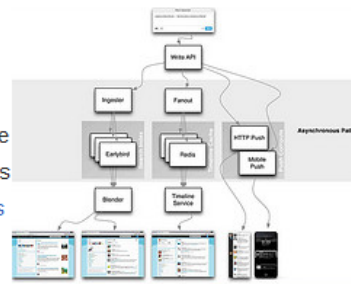
Today Show on television. Over a billion page views are served each month. AOL.com has been a major internet destination since 1996, and still has a strong following of loyal users.

Aol.

The Architecture Twitter Uses To Deal With 150M Active Users, 300K QPS, A 22 MB/S Firehose, And Send Tweets In Under 5 Seconds

MONDAY, JULY 8, 2013 AT 8:54AM

Toy solutions solving Twitter's "problems" are a favorite scalability trope. Everybody has this idea that Twitter is easy. With a little architectural hand waving we have a scalable Twitter, just that simple. Well, it's not that simple as [Raffi Krikorian](#), VP of Engineering at Twitter, describes in his superb and very detailed presentation on [Timelines at Scale](#). If you want to know how Twitter works - then start here.



Topic: [Cloud](#) [Investigate](#)

Follow via: [RSS](#) [Email](#)

The biggest cloud app of all: Netflix

Summary: The largest pure-cloud play service of all is based on Netflix's open-source stack running on Amazon Web Services.



By [Steven J. Vaughan-Nichols](#) for [Networking](#) | April 21, 2013 -- 19:02 GMT (12:02 PDT)

[Follow @sjvn](#)

Topics: [Cloud](#), [Amazon](#), [Linux](#), [Networking](#), [Open Source](#), [Software Development](#)

Netflix, the popular video-streaming service that takes up a [third of all internet traffic](#) during peak traffic hours isn't just the [single largest internet traffic service](#). Netflix, without doubt, is also the largest pure cloud service.

Arkitektur – hvorfor bry seg?

- Gode ting er godt bygget, dvs. de har en god arkitektur:
 - Internett
 - World wide web (W3)
 - Fly
 - Minibanker
 - ...
- God arkitektur er vanskelig
- Dårlig designet programvare kan drepe:
 - Therac 25
 - Ariane 5

Arkitektur – definisjon

- «architectura» fra ἀρχιτέκτων (arkhitékton)
 - *ἀρχι-* (*archi-*), «hoved- eller sjef-», og
 - *τέκτων* (*tékton*), «taktekker, tømrer, murer eller byggmester».

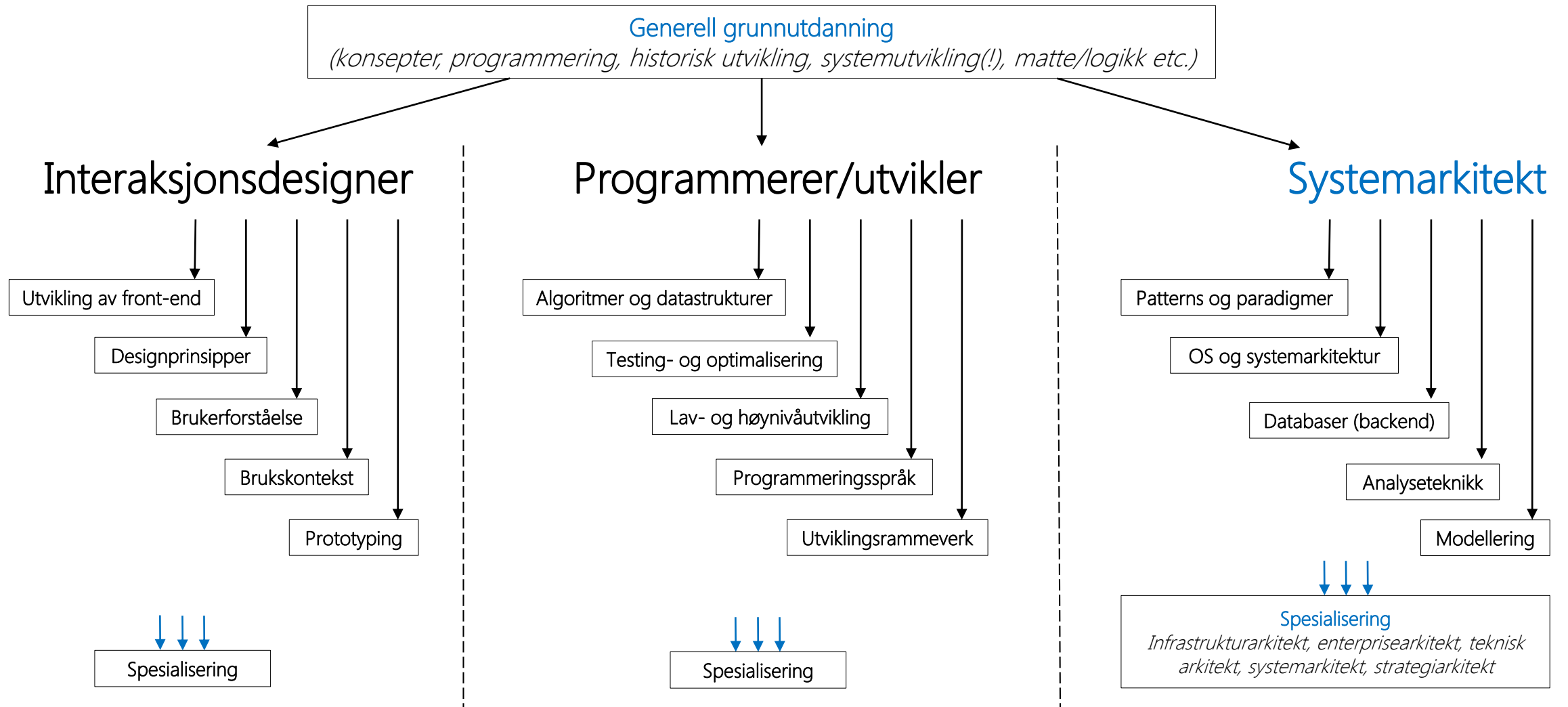
→ «Hovedbyggmester»

- Definisjonen av en systemarkitektur florerer, men fellestrekk:

*«strukturen
eller systemets struktur,
som består av elementer,
deres eksternt-synlige egenskaper,
og forholdet mellom dem»*

(J.P. Sousa, 2011)

Eksempel på utdanningsløp



Systemarkitektur vs. programmering

Arkitektur

- Interaksjonen mellom deler
- Systemhelhetlig fokus
- Deklarativt
- Mer stabile komponenter
- Systemnivå-prestasjon

Programmering

- Implementasjonen av deler
- Lokalt fokus
- Operasjonelt
- Dynamisk allokering
- Algoritmisk prestasjon

"Architect: Someone who knows the difference between that which could be done and that which should be done."

(Larry McVoy)

Analyse vs. design

- Analyse:
 - Spør «hva er problemet?»
 - Hva skjer i gjeldende system?
 - Hva trengs i det nye systemet?
 - Resulterer i en forståelse av:
 - Krav
 - Domeneegenskaper
 - Fokuserer på måten menneskelige aktiviteter gjennomføres

Analyse vs. design

- Design:
 - Undersøker «hvordan vi skal bygge systemet»
 - Hvordan skal det nye systemet fungere?
 - Hvordan kan vi løse problemet som analysen identifiserte?
 - Resulteterer i en løsning på problemet
 - Et fungerende system som imøtekommer kravene
 - *Hardware + software + peopleware*
 - Fokuserer på byggingen av en teknisk løsning
- Analyse og design er ulike aktiviteter
 - Kommer ikke nødvendigvis i rekkefølge

Modeller

Hvordan jobber en arkitekt?

- Artikulering av arkitektoniske visjoner, konseptualisering og eksperimentering med alternative arkitektoniske tilnærminger, utvikling av modeller, samt lage komponent- og grensesnittspesifikk dokumentasjon.
- Validering av arkitektur opp mot kravspesifikasjon og antagelser, forberede dokumenter og forklare arkitekturen til sponsorer og interessenter.

Hva må en arkitekt beherske?

- Avsløre strukturen i systemet, men samtidig gjemme vekk detaljer omkring implementasjon
- Realisere alle bruksmønstre (use cases) og scenarier.
- Adressere krav fra ulike interessenter.
- Håndtere både funksjonelle og kvalitative krav.

(D. Bredemeyer, 2006)

Arkitektoniske modeller

- I et hus har man modeller/tegninger over utforming, rom, elektriske kretser, rørlegging og ventilasjon.
 - Hver modell tilfører et eget **view** over huset
 - Brukes av ulike personer til å måle ulike aspekter og egenskaper ved huset
 - Fungerer som en **beskrivelse** av huset ved konstruksjon, og en **veiledning** ved inspeksjon, renovasjon etc.
- Hvilke strukturer bruker en arkitekt for å gi et eller flere **view** over systemet?

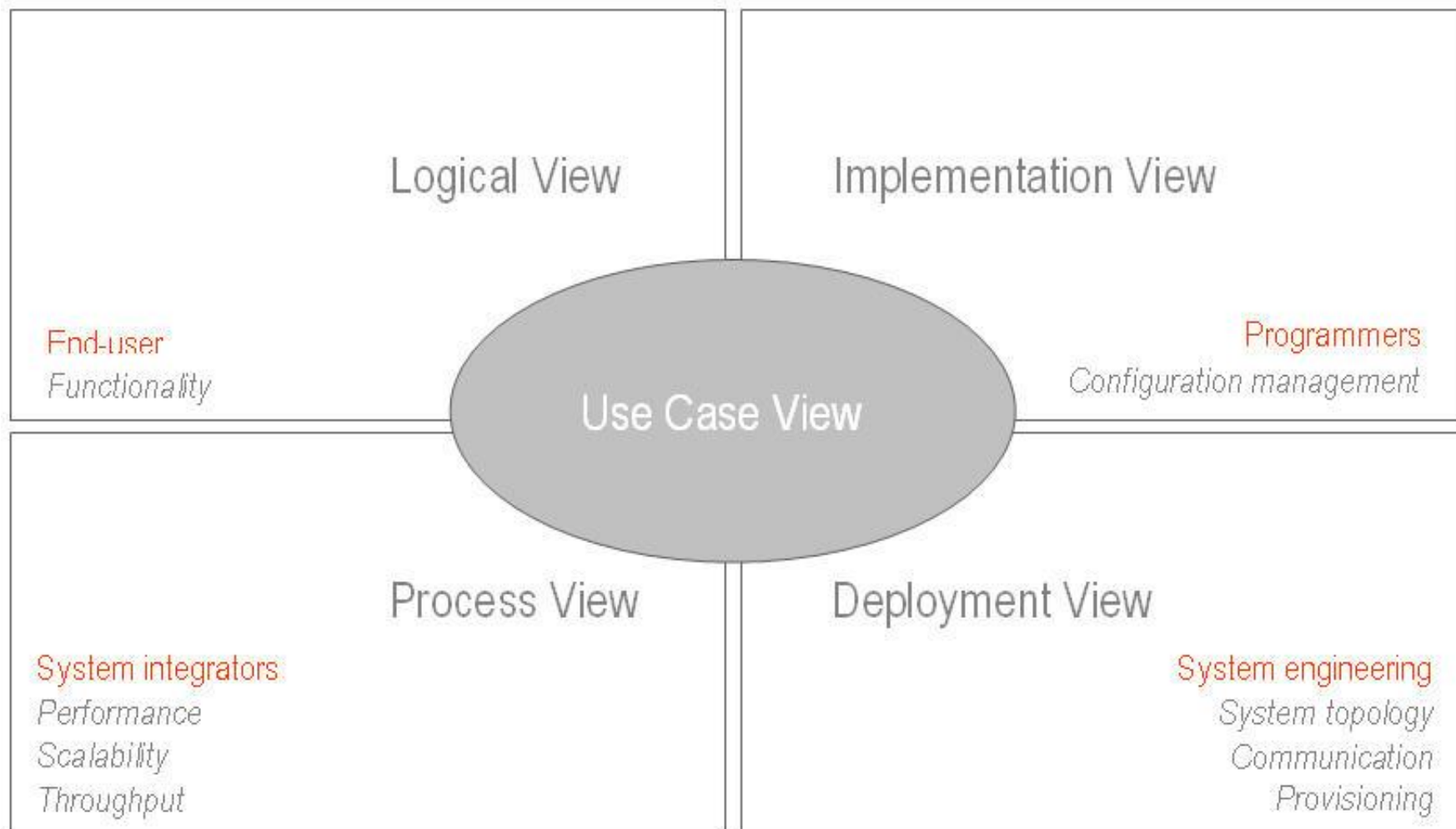


(P. C. Clements, 2002)

4 + 1 view-modellen

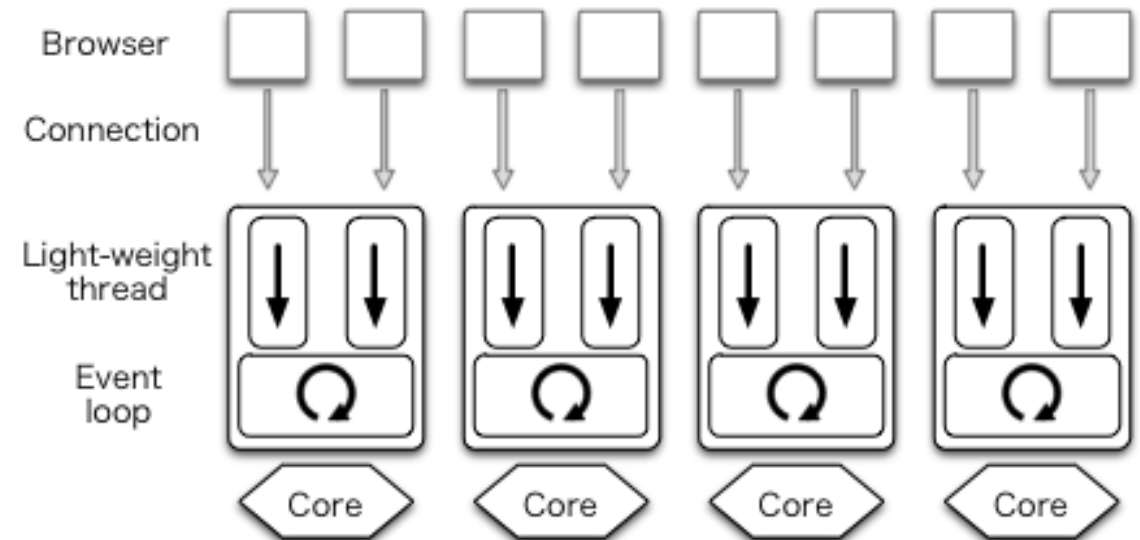
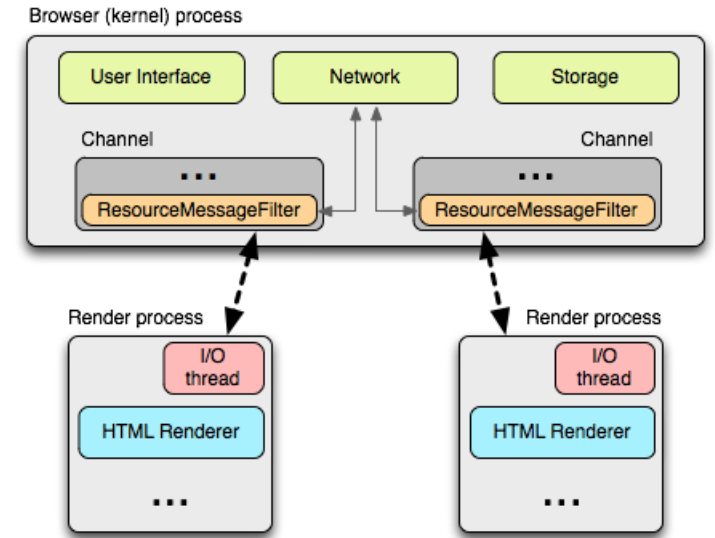
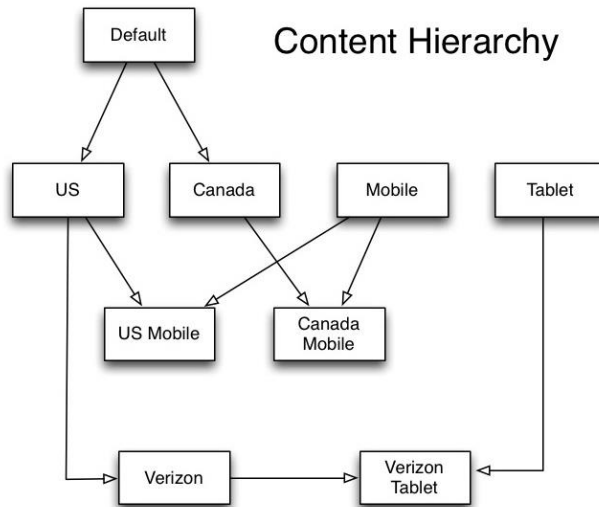
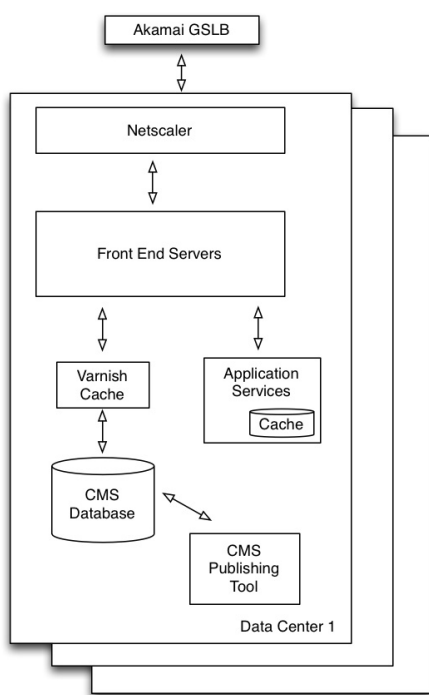
- Brukes til å illustrere systemet fra ulike perspektiver
- **Logisk view**
Viser nøkkelabstraksjoner i systemet som objekter eller objektklasser (*klassediagram, sekvensdiagram*)
- **Prosessview**
Viser hvordan (ved kjøretid) et system er sammensatt av interagerende prosesser (*aktivitetsdiagram*)
- **Utviklingsview**
Viser hvordan programvaren er dekomponert for utvikling (*pakkediagram (UML Package diagram)*)
- **Fysisk view**
Viser systemets maskinvare, samt hvordan programvarekomponenter er distribuert på tvers av prosessorer (*Deployment diagram*)
- Bruker relevante bruksmønstre (use cases) eller scenarier for å knytte det sammen (+1).

4 + 1 view-modellen



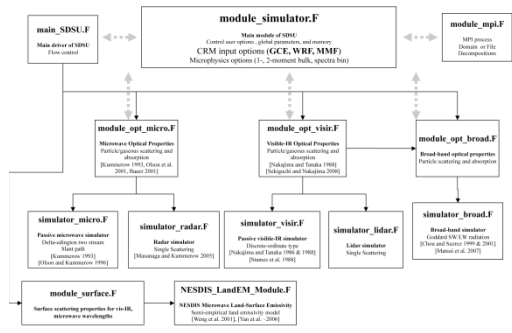
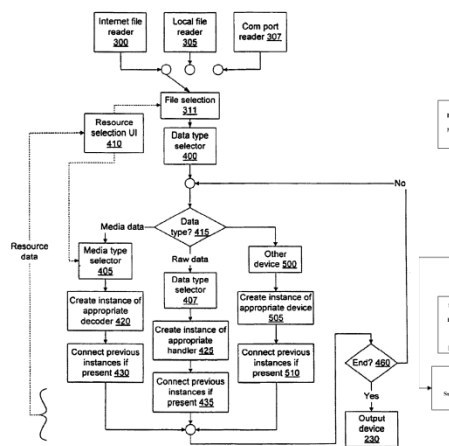
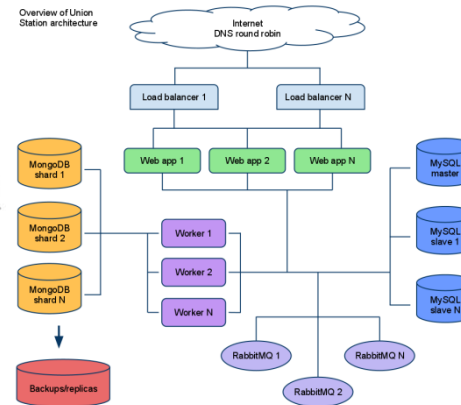
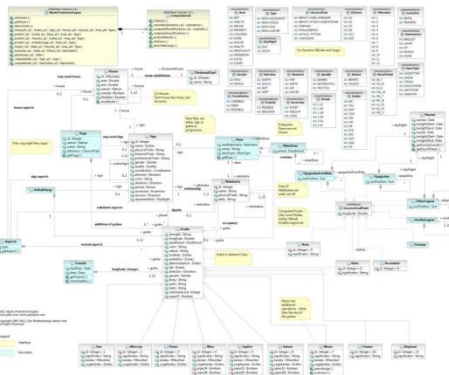
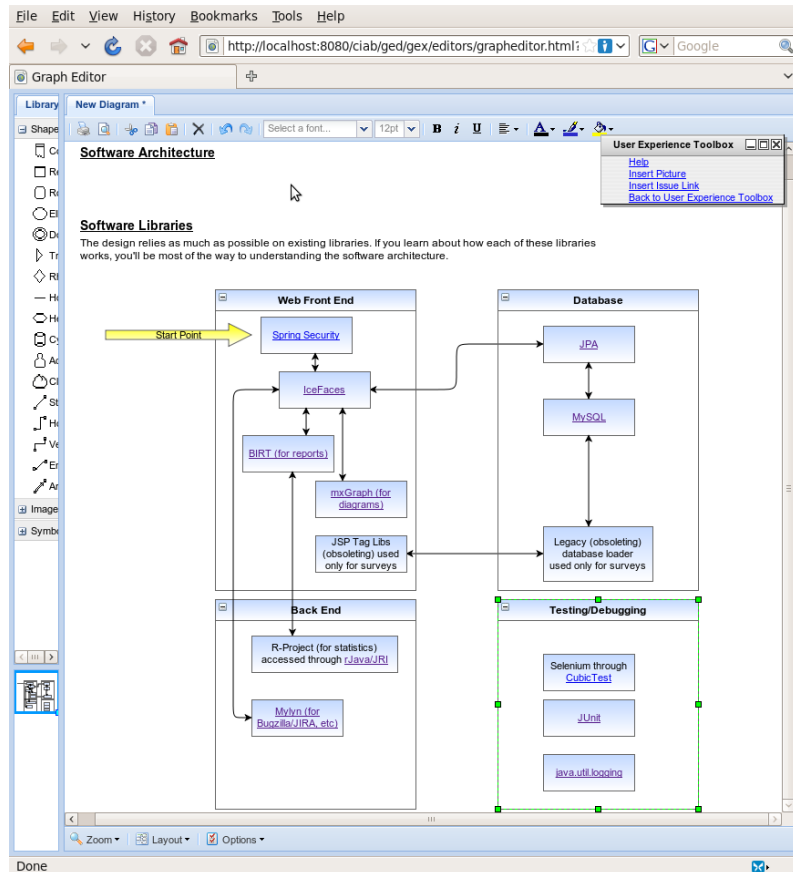
(J. Parnitzke, 2009)

Hvordan ser modellene typisk ut?



<http://highscalability.com/blog/2014/2/17/how-the-aolcom-architecture-evolved-to-99999-availability-8.html>
<http://www.aosabook.org/en/posa/warp.html>

Hvordan ser modellene typisk ut?

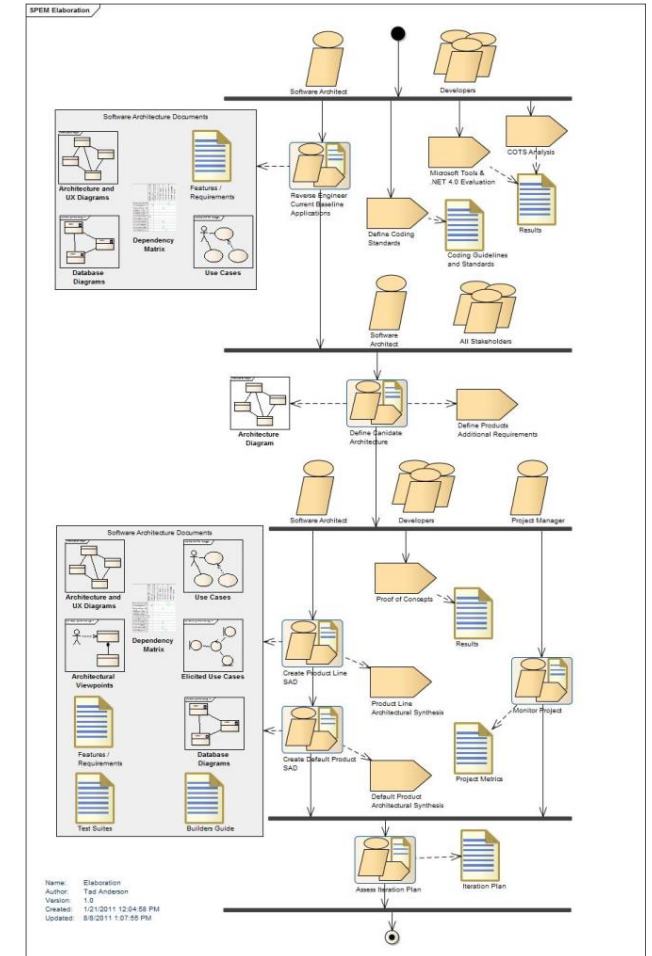


<https://open.jira.com/wiki/download/attachments/4292797/DiagramEdit-SoftwareArchitecture.png?version=1&modificationDate=1255794053637>
http://cloud.gsfc.nasa.gov/uploads/sdsu_Figure2.png
<http://www.freepatentsonline.com/6742176-0-large.jpg>
<http://blog.phusion.nl/wp-content/uploads/2011/03/UnionStationarchitecture.png>

Hvordan ser modellene typisk ut?

- Som oftest en uformell sammenhengende prosatekst med en tilhørende figur
- Figuren består av bokser og linjer
- Still krav til intuisjon hos leser
- Sjeldent formelt
- Lite presisjon

http://softwareprocessengineering.com/_ple/Elaboration.jpg



Systemarkitektur

- Tilfører en [designplan](#) for et system:
 - Blåkopi
 - Innebærer *formålet*
- Er en abstraksjon som hjelper med å håndtere kompleksiteten i et system
- Systemarkitektur er begrenset av:
 - Mangel på standardiserte måter å representere arkitektur
 - Mangel på analytiske metoder som kan forutsi hvorvidt en [arkitektur](#) vil resultere i en [implementasjon](#) som imøtekommer [kravspesifikasjonen](#)

[ISO/IEC/IEEE 42010:](#)
Systems and software engineering - Architecture description.

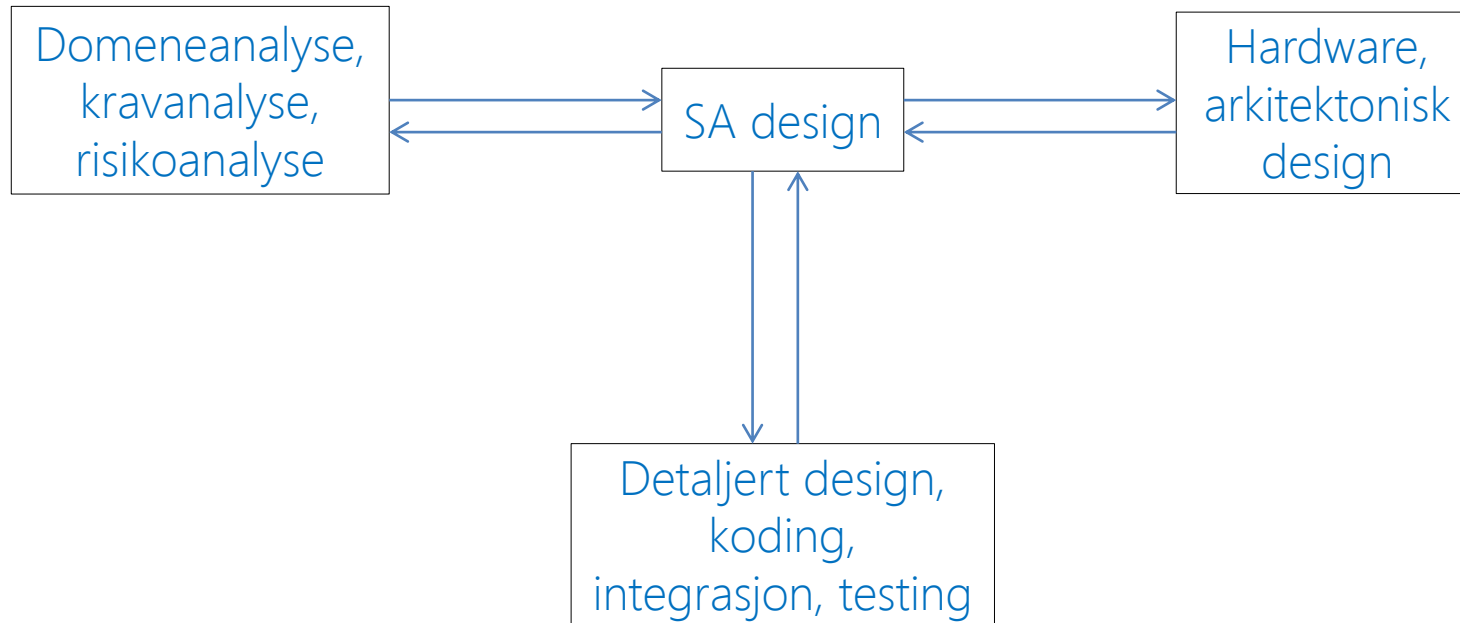
Designplaner er dokumentasjon. Dokumentasjon er viktig også for en arkitekt!

→ [Se eksempel](#)

Arkitektur som designplan

- Strukturell plan som beskriver:
 - Systemets elementer
 - Hvordan de passer sammen
 - Hvordan de samhandler for å imøtekomme kravene
- Brukes som blåkopi i utviklingsprosessen
- Brukes til å forhandle om systemkrav
- Bruks til å kontrollere forventninger til:
 - Kunder
 - Brukere
 - Markedsføring

Arkitektur som designplan

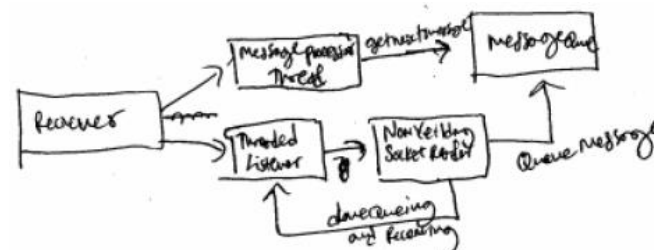
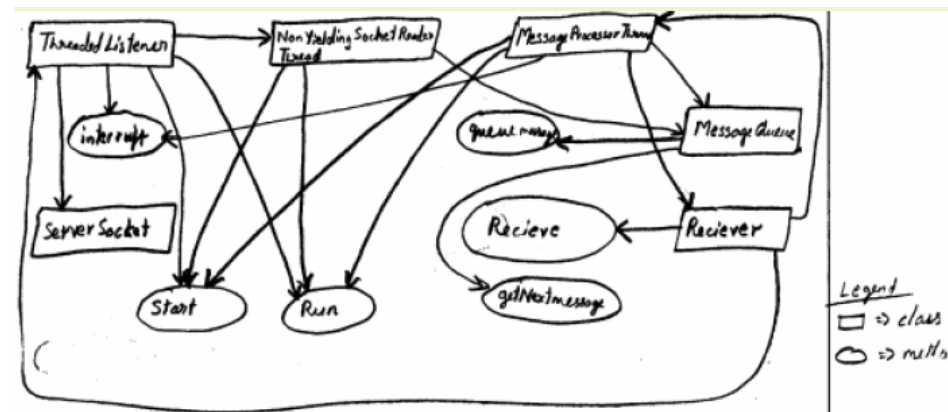
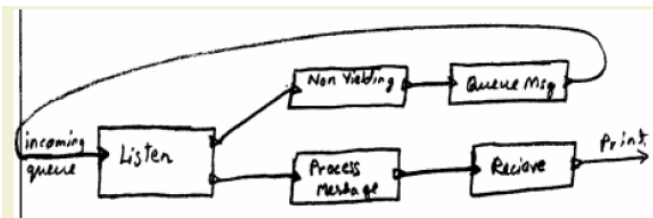
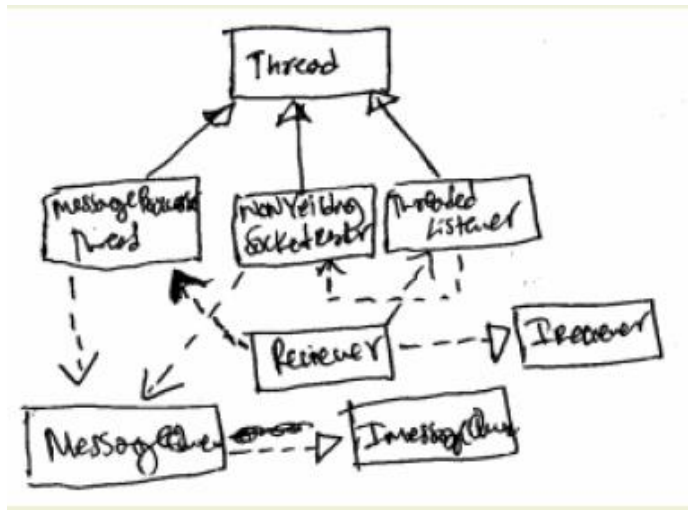


(L. Petre, 2010)

Utfordringer ved arkitektur

Hvorfor er det vanskelig?

- Bygninger og andre ingeniørdisipliner er synlige, software er «usynlig»
- Arkitekter kan produsere helt forskjellige tegninger for samme stykke kode



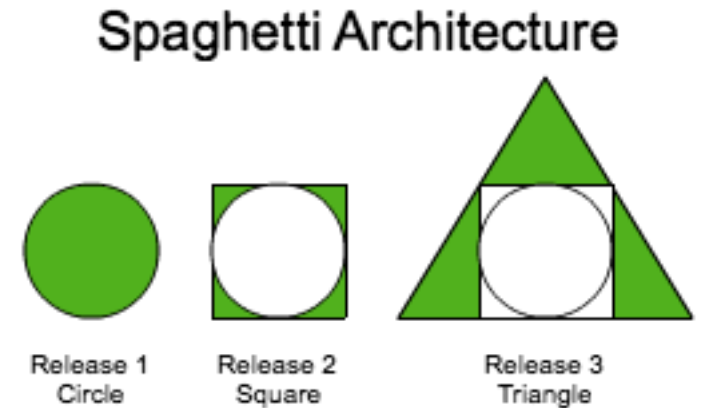
(J.P. Sousa, 2011)

Utfordringen

- Utvikle et system «arkitektonisk»:
 - Bruk etablert arkitektonisk designekspertise
 - Bygg systemer ved å sette sammen eksisterende deler
 - Bruk standarder for et system
 - Sikre at systemet har ønskede egenskaper
 - Reduser prosess- og utviklingskostnader
- Utfordringen er å gjøre systemarkitektur til ingeniørdisiplin
- ... fra ad hoc definisjoner til etablerte og nedfelte prinsipper

Kriterier under utvikling

- Kriteriene har forandret seg:
 - Maskinvare (hardware) har blitt forbedret og mer kostnadseffektivt
 - Behovet for programvare har eksplodert
 - Gjelder å være første programvaren på markedet
 - *Hvordan definere krav for nye produkter og implementere programvaren raskt og billig?*
- Har utviklingsprosjektet en god systemarkitekt, forstått av eksterne interessenter og interne utviklere?



Figur: <https://thomashunter.name/blog/spaghetti-architecture-and-the-importance-of-starting-over/>

Når bør arkitektoniske beslutninger tas?

Typisk leveringsmekaniske for produktet	Kostnader ved utvikling eller oppgradering	Brukerinvolvering i oppgraderingen	Når man bør foreta en arkitektonisk beslutning
Software as a Service (SaaS)	Ubetydelige	0	Så sent som mulig
Bokset (sendt) programvare	Minimale	Total	Så sent som mulig
Programvare oppgraderbar etter utplassering	Lave til håndterbare	Minimal	Siste ansvarlige øyeblikk, men aldri senere
Maskinvare eller annet produkt som er vanskelig å oppgradere	Høye	Signifikant	Så tidlig som mulig grunnet programrisiko (utvikling og levering)

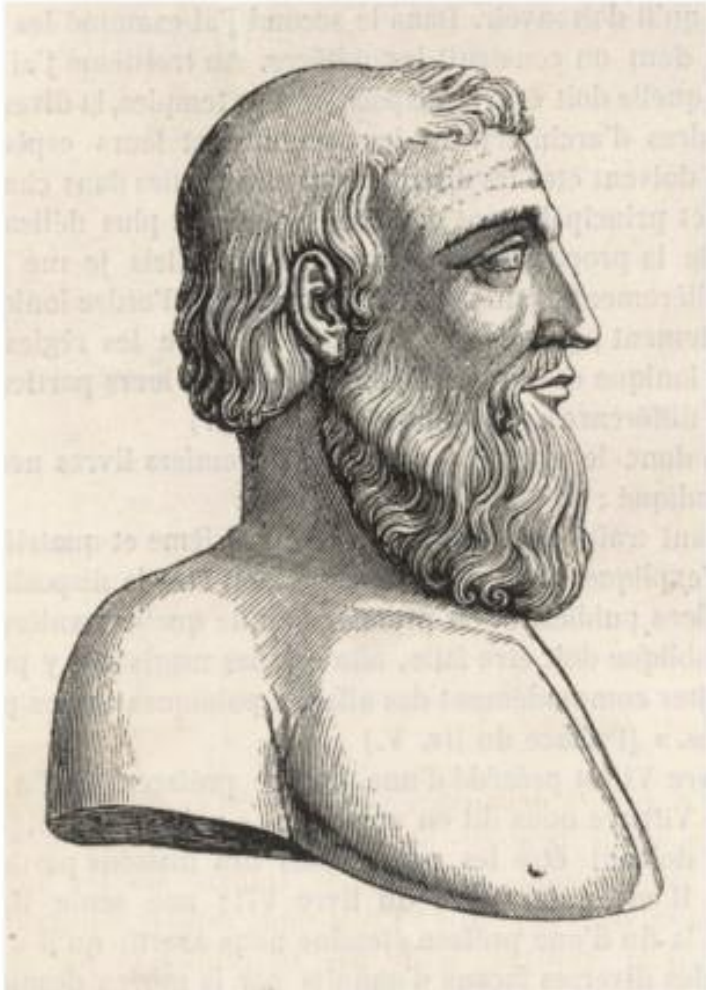
(J. Rothman, 2011)

Arkitektur: designnivåer

- Arkitektonisk design (*høy-nivå design*)
 - Arkitektur – helhetlig struktur: hovedmoduler og deres forbindelser
 - Design som dekker de viktigste use casene til systemet
 - Adresserer de viktigste ikke-funksjonelle kravene
 - Vanskelig å forandre
- Detaljert design (*lav-nivå design*)
 - Den indre strukturen til hovedmodulene
 - Kan i noen tilfeller inkludere programmeringsspråket i beslutningen
 - Detaljert nok til å bli implementert i ønsket programmeringsspråk

Sommerville bruker begrepene «*architecture in the small*» og «*architecture in the large*» for å illustrere et tilsvarende poeng.

Hva er god arkitektur?



Hva er god arkitektur?

Bygninger

- Holdbarhet (*firmitas*)
 - Stabile
 - Varige
 - Motstandsdyktige
- Skjønnhet (*venustas*)
 - Balanserte proposjoner
 - Vakkert
- Nytte (*utilitas*)
 - Brukbare
 - Formålstjenlige

(Vitruvius, «Ti bøker om arkitektur», 23-27 f.Kr)

Systemer

- Pålitelig
- Gir god ytelse
- Feil-tolerant
- Vedlikeholdbar
- Rask i responsen
- Forståelig
- ...

God systemarkitektur

- En systemarkitekt bør etterstrebe **god arkitektur**:
 - Når systemet er implementert i henhold til arkitekturen så imøtekommer det kravene og ressursbudsjetter
- Det er mulig å implementere et system i henhold til arkitekturen
- Resultatet av en konsekvent bruk av prinsipper og teknikker gjennom alle fasene i prosjektet
- Robust når det skal gjøres (uunværlige) endringer
- Kilde til veiledning gjennom hele systemets levetid
- Gjenbruk av etablert og nedfelt ingeniørkunnskaper

Ikke godt nok når:

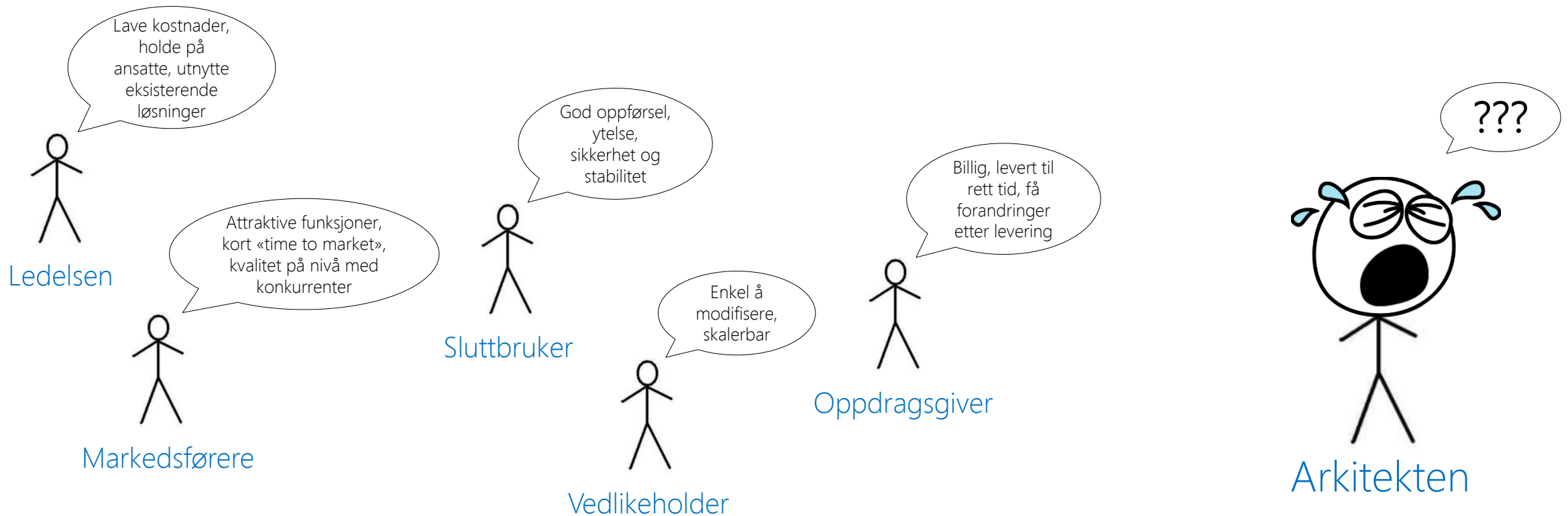
- Ikke eksplisitt
- Ikke alt-omfattende
- Ikke konsekvent
- Ikke forståelig

Viktige spørsmål for en systemarkitekt

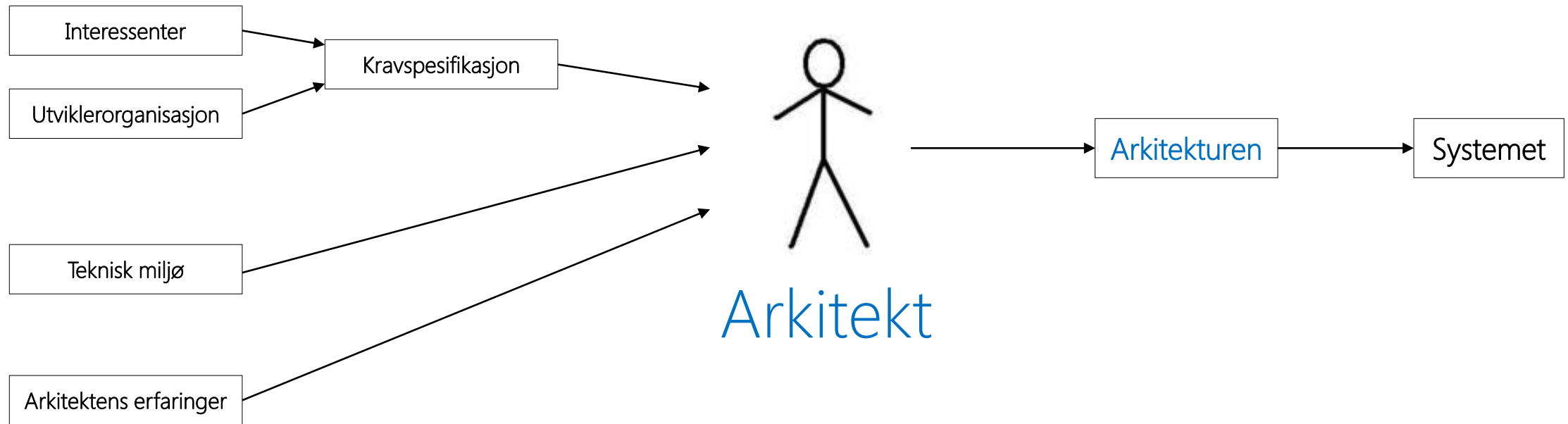
- Finnes det en generisk applikasjonsarkitektur som kan brukes?
- Hvordan vil systemet distribueres?
- Hvilken arkitektonisk stil er passende?
- Hvilken tilnærming vil bli brukt til å strukturere systemet?
- Hvordan vil systemet bli oppdelt i moduler?
- Hvilken kontrollstil bør vi bruke?
- Hvordan skal den arkitektoniske utformingen evalueres?
- Hvordan skal arkitekturen dokumenteres?

Hvem påvirker en arkitekt?

- **Enkelt svar:** alle interessenter stiller krav som en arkitekt må forholde seg til.
- Eksempler på krav fra interessenter som direkte påvirker arkitektens spillerom.



Hva påvirker en arkitekt?



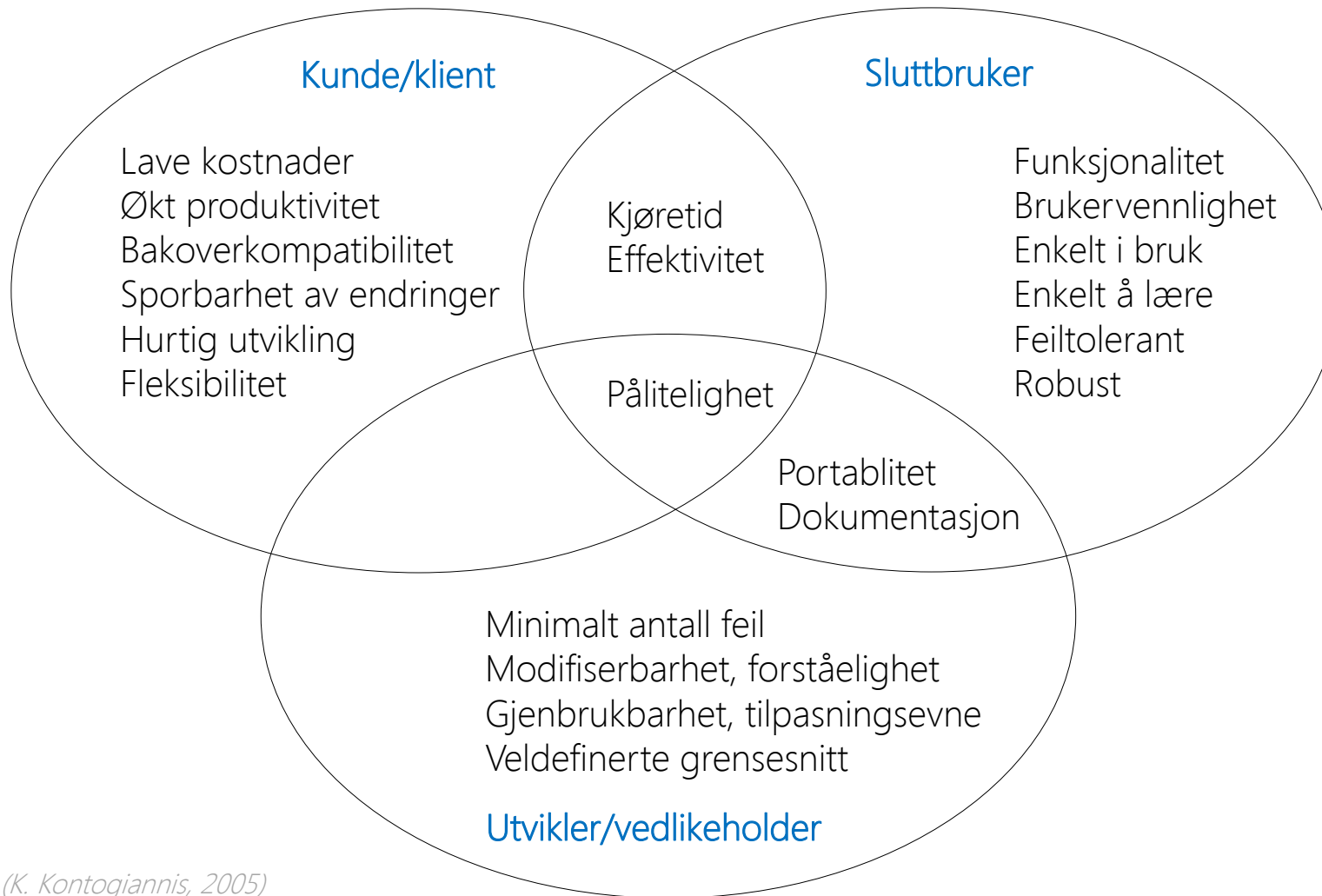
(P. C. Clements, 2002)

Hva kan en arkitekt påvirke?

- **Strukturen på utviklerorganisasjonen**
 - *Kort sikt:* Arbeidsfordeling fordeles ofte rundt de ulike arkitektoniske komponentene i et system under utvikling.
 - *Lang sikt:* Mange liknende systemer → bedriftens enheter reflekterer komponenter (f.eks. databaseenhet, OS-enhet).
- **Bedriftsmål hos utviklerorganisasjonen**
 - Å være kjent for utviklingen av et spesielt type system gir et markedsføringsmiddel.
 - Arkitekturen gir en mulighet til ytterligere markedsmuligheter og samarbeid.
- **Kundens krav til systemet**
 - Nye kunder vil spørre etter egenskaper i liknende systemer.
 - Kunder vi tilpasse sine krav på grunnlag av tilgang på eksisterende systemer.
- **Andre faktorer:**
 - Arkitektens egen erfaring og selvutvikling
 - Teknisk miljø (trelags databasearkitektur)
 - Arkitekturen i seg selv

(P. C. Clements, 2002)

Forholdet mellom ulike designmål



(K. Kontogiannis, 2005)

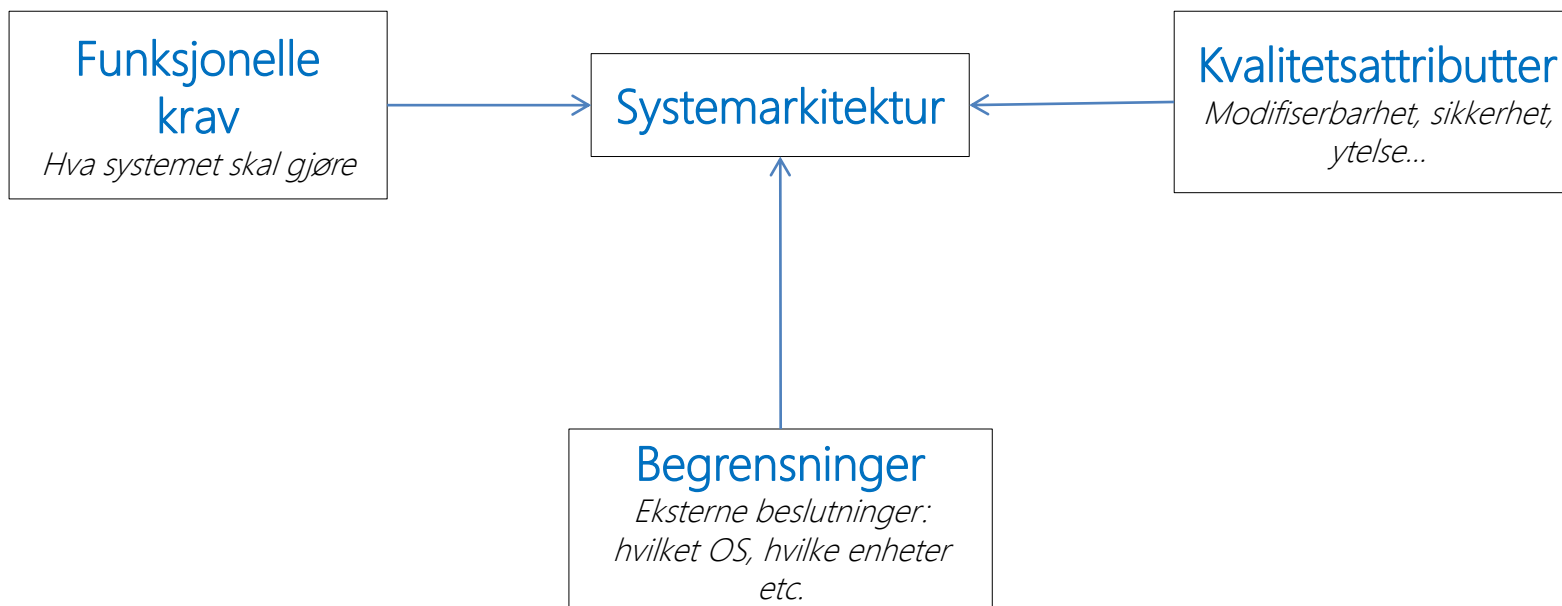
Typisk design trade-offs

- Funksjonalitet vs brukbarhet
- Kostnad vs robust
- Effektivitet vs portabilitet
- Rask utvikling vs funksjonalitet
- Kostnad vs gjenbrukbarhet
- Bakoverkompatibilitet vs leslighet

Arkitektoniske drivere

Arkitektoniske drivere

- Hva er med på å forme arkitekturen?



(J.P. Sousa, 2011)

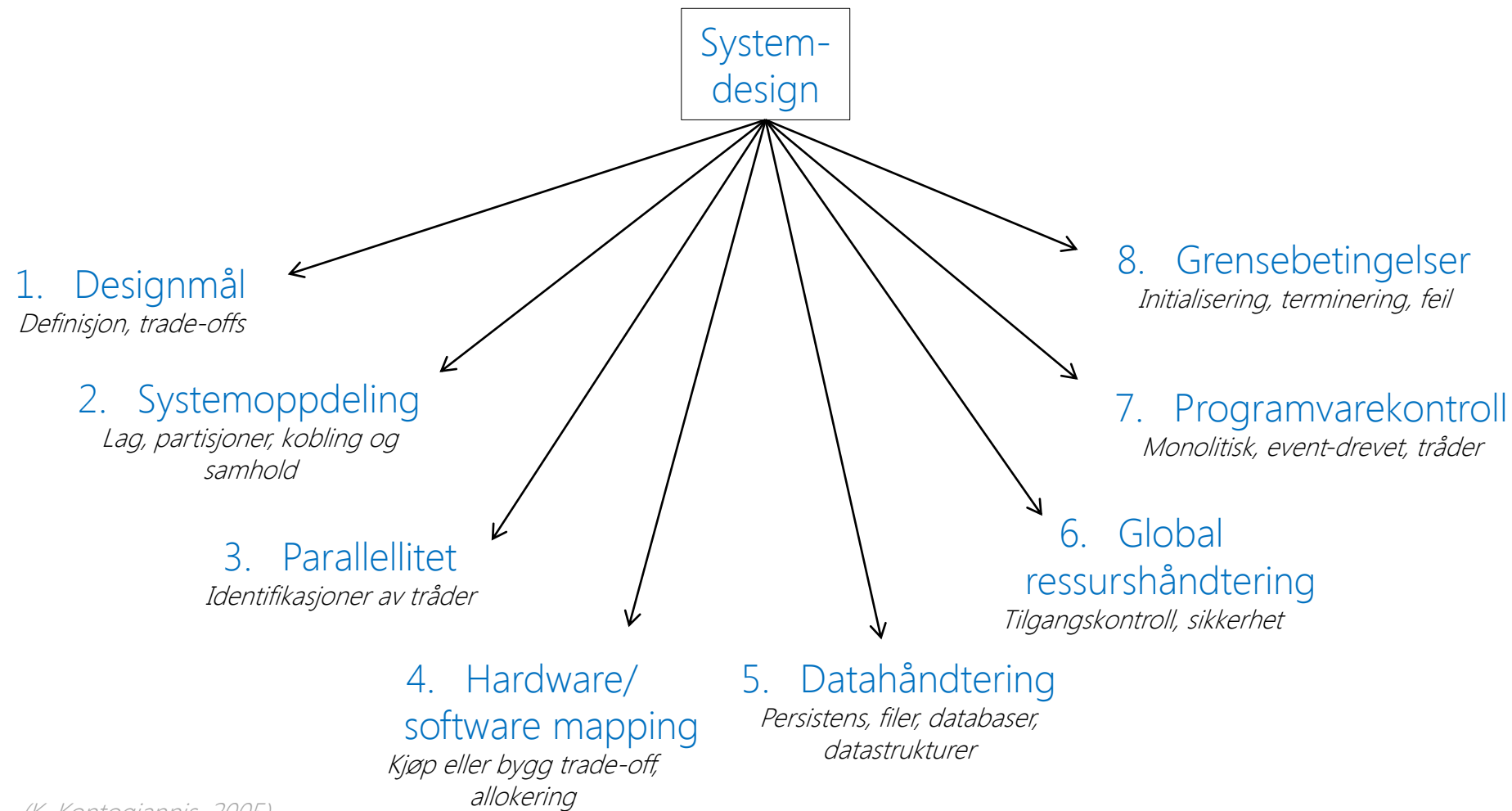
Kvalitetsattributter

- Hvilke kvaliteter skal bli en del av systemets arkitektur?
 - Kvalitetsattributter tas i betraktning under design, implementasjon og utrulling
- Slike kvaliteter er som regel *ikke-funksjonelle*
- Avhenger som regel i stor grad av struktur → *ulik struktur fremmer ulike kvaliteter*
- **Systemkvaliteter:** tilgjengelighet, modfiserbarhet, ytelse, sikkerhet, testbarhet, brukbarhet
- **Businesskvaliteter:** «time-to-market», kostnad
- **Arkitektoniske kvaliteter:** forståelighet, konseptuell integritet

Kvalitetsattributter (ikke-funksjonelle krav)

- Ikke-funksjonelle krav
 - Ytelse
 - Kjøretid
 - Tilgjengelighet
 - Gjenopprettbarhet
 - Sikkerhet
 - Skalerbarhet
 - ...
- Brukbarhet
 - Lærbarhet
 - Konfigurerbarhet
 - Dekning
 - ...
- Vedlikeholdbarhet
 - Utvidbarhet
 - Testbarhet
 - Portabilitet
 - ...

Systemdesign



(K. Kontogiannis, 2005)

Liste med mål for design



- List of design goals:
 - Reliability
 - Modifiability
 - Maintainability
 - Understandability
 - Adaptability
 - Reusability
 - Efficiency
 - Portability
 - Traceability of requirements
 - Fault tolerance
 - Backward-compatibility
 - Cost-effectiveness
 - Robustness
 - High-performance
 - Good documentation
 - Well-defined interfaces
 - User-friendliness
 - Reuse of components
 - Rapid development
 - Minimum # of errors
 - Readability
 - Ease of learning
 - Ease of remembering
 - Ease of use
 - Increased productivity
 - Low-cost
 - Flexibility

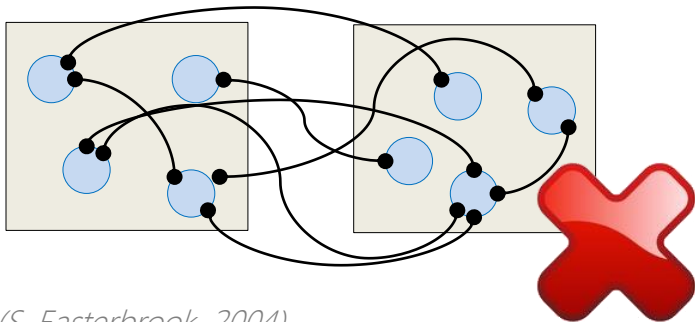
(K. Kontogiannis, 2005)

Arkitektoniske stiler

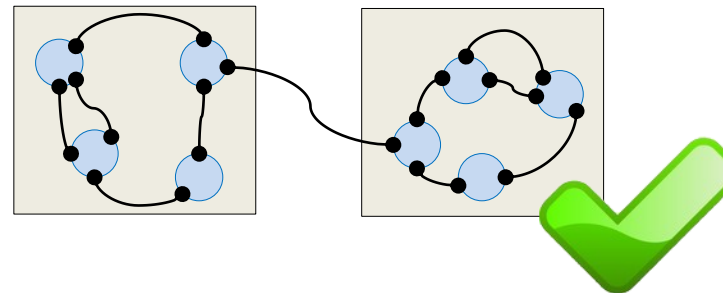
Kobling og samhold

Må kunne forklare hvorfor minimal kobling og maksimalt samhold mellom moduler er en ønskelig egenskap ved et systems arkitektur.

- Kalles for *coupling* og *cohesion* på engelsk
- Arkitektoniske byggestener  
- En god arkitektur:
 - Minimerer *koblinger* mellom moduler (*løs kobling*)
 - **Målsetning:** moduler behøver ikke vite mye om hverandre for å interagere
 - Lavt antall koblinger gjør fremtidige endringer enklere
 - Maksimere *samholdet* til hver modul
 - **Målsetning:** innholdet i hver modul henger sterkt sammen
 - Høyt samhold gjør en modul enklere å forstå



(S. Easterbrook, 2004)



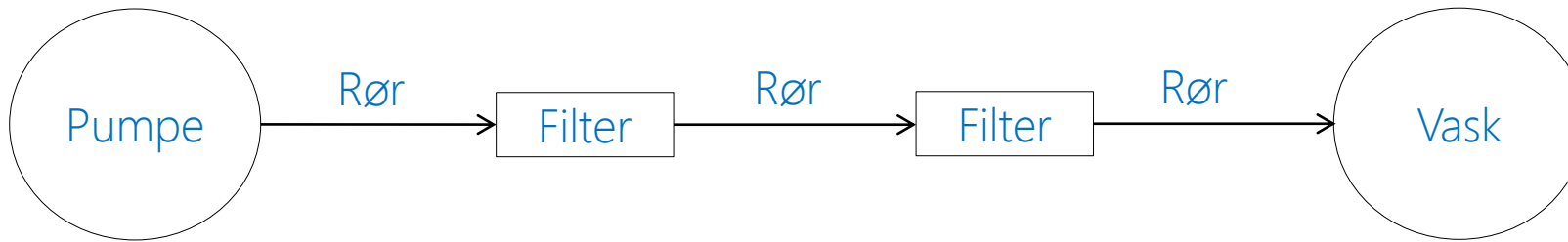
Arkitektonisk stil

- Kalles for *architectural pattern* på engelsk
- Skiller seg fra *design pattern* ved at man har et bredere perspektiv
- Standardiserte måter for å representere systemarkitektur
- De ulike mønstrene har ulike tilnærminger og ulikt fokus

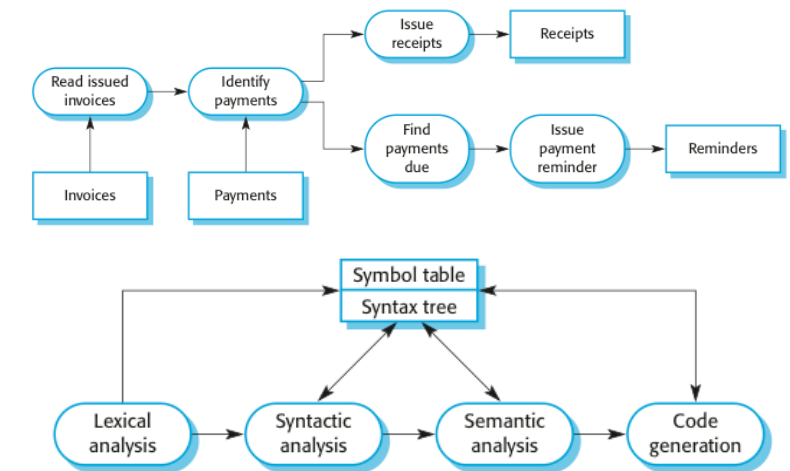
Kategori	Arkitektonisk stil
Kommunikasjon	<i>Service-Oriented Architecture (SOA), Message Bus, Pipes and Filters</i>
Utrulling	<i>Klient/server, 3-Tier, N-Tier</i>
Domene	Domene-drevet design
Struktur	Komponent-basert-, Objekt-orientert- og <i>lagdelt arkitektur, MVC</i>

Pipe and filter

Må kunne forklare tanken bak «pipe and filter», samt komme med eksempler på bruk av denne arkitektoniske stilen.



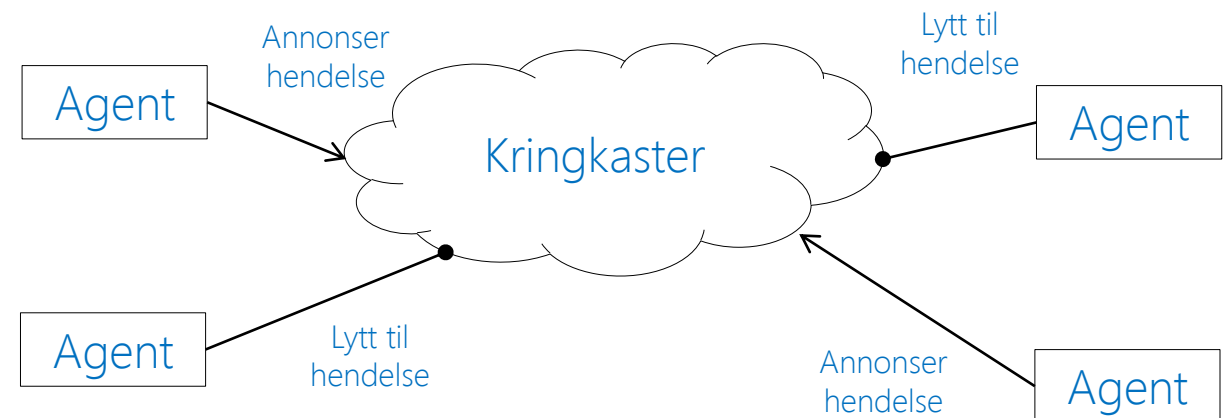
- Komponenter (**filter**) transformerer eller filtrerer data før den sendes videre via tilkoblingen (**rør**) til andre komponenter
- Filtrene fungerer samtidig, ofte implementert i parallell
- Filtrene behøver ikke å vite noe om hva de er koblet til
- Systemets oppførsel er definert av filtrenes oppførsel
- **Eksempel:**
 - UNIX Shell: pipe
 - Kompilatorer: leksikalsk analyse → parsing → semantisk analyse → kodegenerering



Event-driven architecture (EDA)

Må kunne forklare hvordan EDA skiller seg fra andre arkitektoniske stiler, og hvilke fordeler denne har ovenfor andre stiler.

- Arkitekturen er bygget rundt real-time hendelser ([events](#))
- [Agenter](#) annonserer og lytter til hendelser og reagerer på det
- Annonsøren av en hendelse behøver ikke å vite hvem som lytter eller håndterer den
- Støtter gjenbruk og utvikling av systemet (nye agenter kan enkelt tilføyes)
- *Kan kombineres med SOA*
- **Eksempler:**
 - Debugging-systemer (med lyttere, breakpoints)
 - DBMS (for integritetssjekk av data)
 - GUI

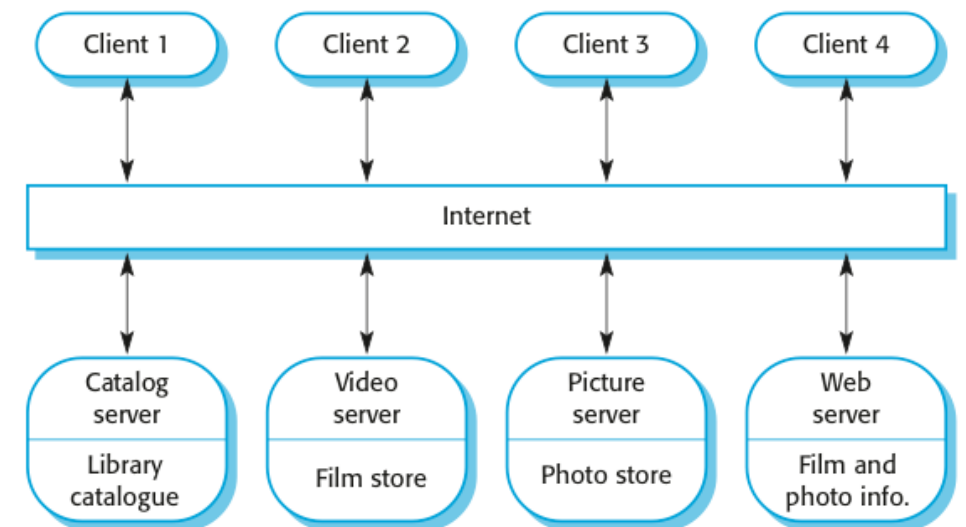
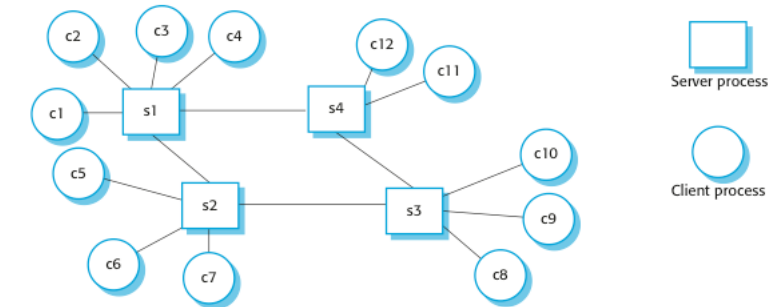


(S. Easterbrook, 2004)

Klient-server arkitektur

Må kunne redegjøre for hvorfor denne typen arkitektur egner seg bedre for parallell aksessering, samt hvilke sårbarheter som typisk inngår i denne typen arkitektur.

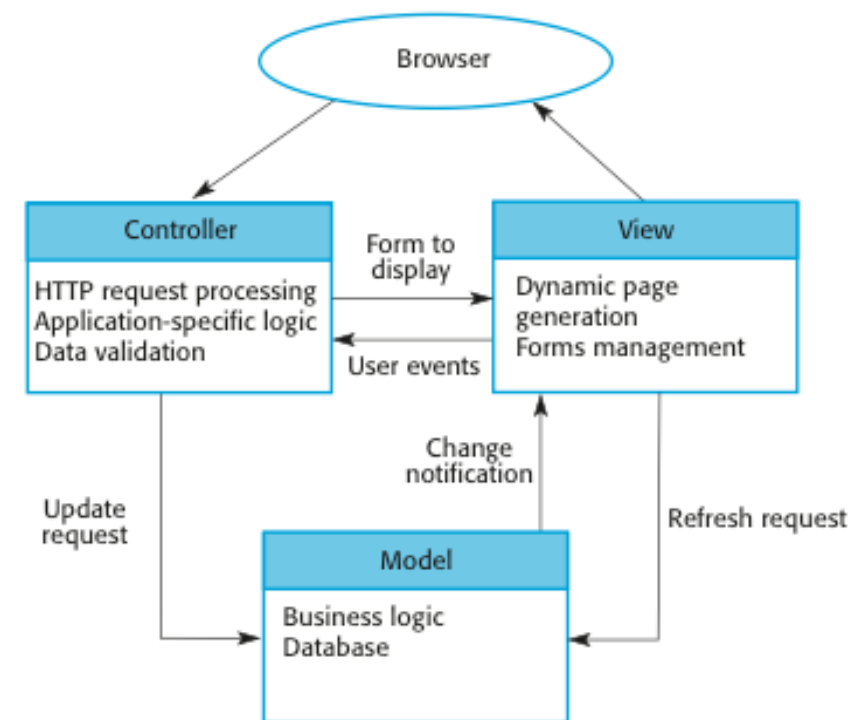
- Systemets funksjonaliteter er organisert i tjenester
- Hver tjeneste leveres fra en egen server
- Brukes typisk når delt data skal aksesseres fra en rekke steder samtidig
- Siden de serverne som er mest belastet kan speiles kan dette skalere relativt lett
- Servere kan distribueres over nettverket
- Lite redundans → hver server er sårbar mot f.eks. Dos/DDos-angrep
- Brukes også her på UiO



MVC

Må være i stand til å forklare hva som inngår i de tre ulike logiske komponentene som inngår i MVC, samt hvordan disse samhandler.

- «Model-View-Controller»
- Separerer presentasjon og interaksjon fra systemets data.
- Strukturert i tre logiske komponenter som samhandler.
- **Model:** håndterer systemets data og tilhørende operasjoner på dataen.
- **View:** definerer og håndterer hvordan dataen presenteres for brukeren.
- **Controller:** håndterer brukerinteraksjon (tastetrykk, museklikk etc.) og sender instruksjoner til View og Model.



Lagdelt arkitektur

Må kunne påpeke fordeler og ulemper ved å bruke en lagdelt arkitektur, samt gi en enkel forklaring på hva det vil si at en arkitektur er lagdelt.

- Organiserer systemet i lag med spesifikk funksjonalitet tilhørende hvert lag
- Lagene former et abstrakt hierarki
- Hvert lag kan i prinsippet «erstattes» uten at det påvirker resten av systemet
- Som regel 2-, 3- eller 4-delt lagdeling
- Skiller mellom fysisk og logisk lagdeling
- Kan virke kunstig med slik inndeling
- Vil ofte være noe tregere da tjenestespørringer må tolkes flere ganger av ulike lag

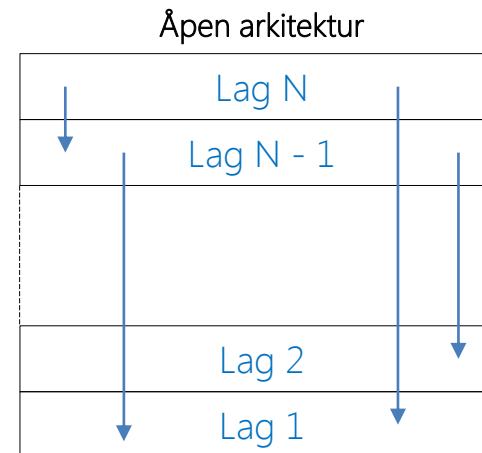
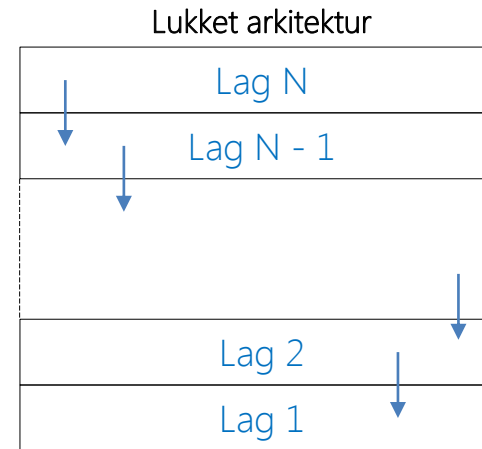


(<http://img.ehowcdn.com/article-new/ehow/images/a08/1v/v3/decorate-tier-square-wedding-cake-800x800.jpg>)

Lagdelt arkitektur

Må kunne forklare hva som menes med en henholdsvis lukket og åpen arkitektur når man har en lagdelt arkitektur, samt fordeler og ulemper ved disse to typene.

- Lukket arkitektur:
 - Kommunikasjon foregår kun mellom to nabolag
 - Minimerer avhengighet mellom lag og reduserer innvirkningen av endringer
 - Endringer på et lag påvirker maks to andre lag
- Åpen arkitektur:
 - Hvert lag kan kommunisere med alle lag som ligger lavere
 - Mer kompakt kode ettersom lavereliggende funksjonalitet kan brukes direkte
 - Bryter innkapslingen av lag, og øker dermed avhengigheten mellom lag



(S. Easterbrook, 2004)

Lagdelt arkitektur

Må kunne gi eksempler på henholdsvis tolags-, trelags-, og firelags arkitektur, samt vise hvordan enkelte lag kan være partisjonert.

- **2-lags arkitektur:**
 - Applikasjonslag
 - Databaselag
 - F.eks. en enkel klient-server modell
- **3-lags arkitektur:**
 - Skiller ut business-logikk som eget lag
 - Hjelper til med å gjøre UI- og databaselagene lettere modifiserbare
- **4-lags arkitektur:**
 - Skiller applikasjonen fra domeneentiteter som brukes
- **Partisjonert 4-lags arkitektur:**
 - Identifiserer ulike applikasjoner og bruksområder

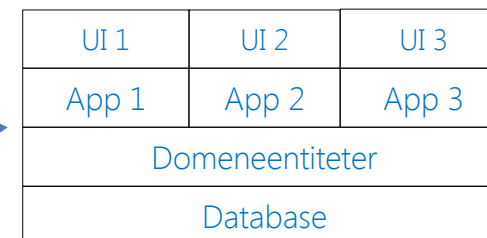
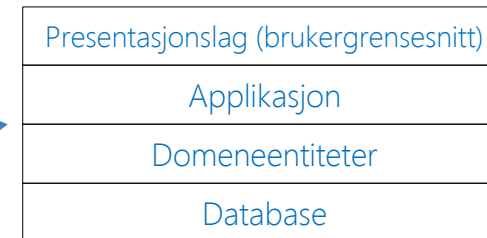
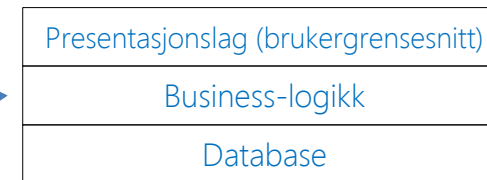


Figure 6.6 A generic layered architecture

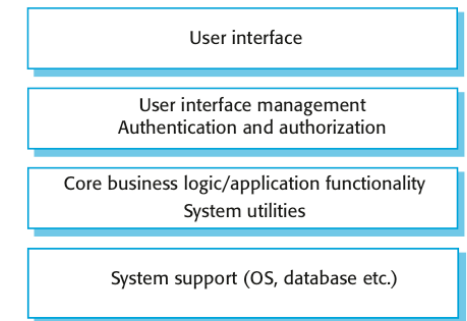
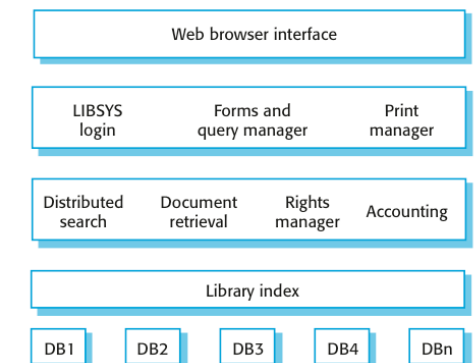


Figure 6.7 The architecture of the LIBSYS system



Lagdelt arkitektur

Må kunne skille mellom fysisk og logisk arkitektur, samt gi et eksempel på hvilke deler av en gitt arkitektur som tilhører henholdsvis logisk og fysisk arkitektur.

- **Fysisk arkitektur** gir en oversikt over de ulike maskiner og enheter, samt hva som kjører på dem
- **Logisk arkitektur** gir en oversikt over hvilke komponenter som inngår i programvaren

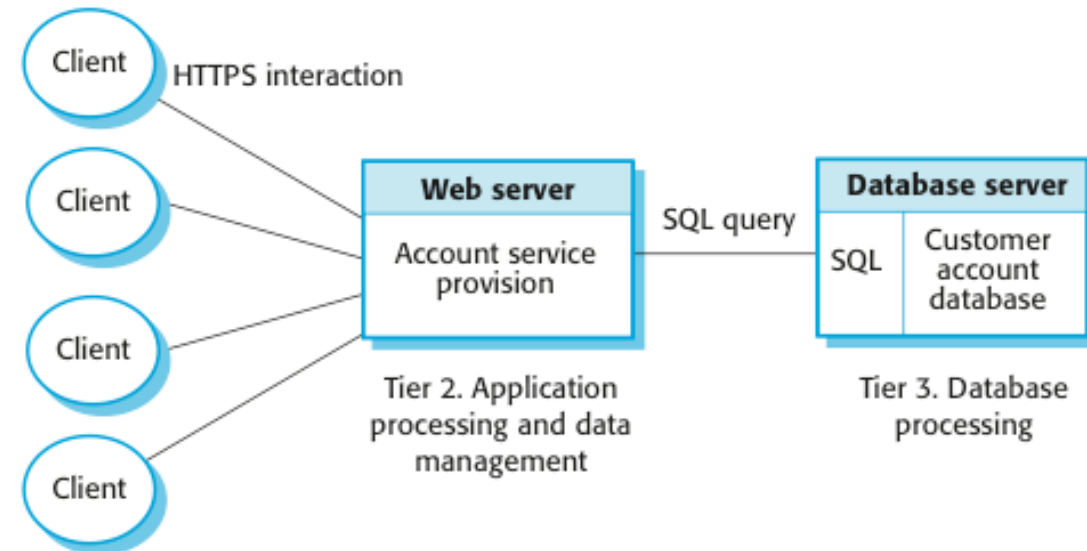
Eksempel på trelags logisk arkitektur

Presentasjonslag
Business-logikk lag
Datalag

Eksempel på firelags fysisk arkitektur

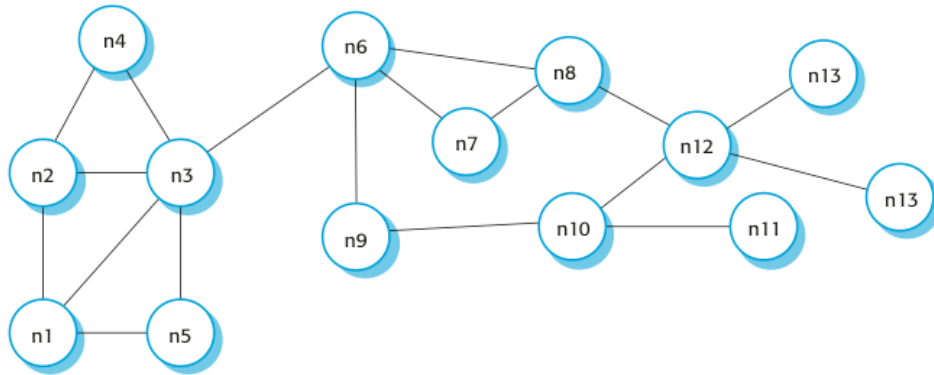
Klient
Webserver
Applikasjonserver
Database

Tier 1. Presentation

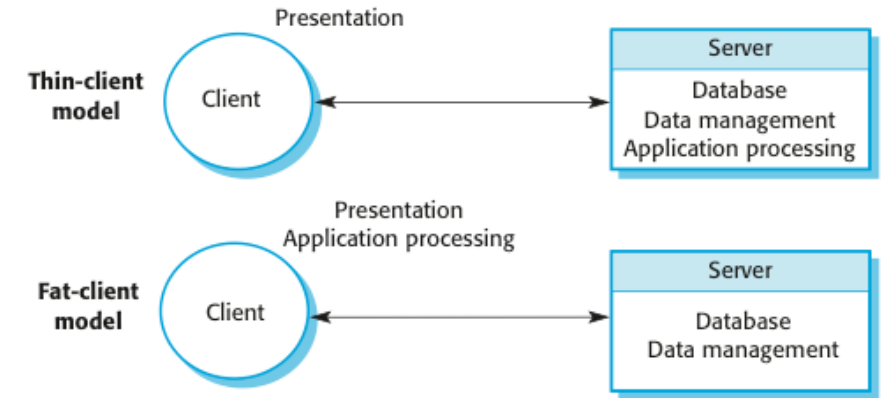


Eksempler på andre arkitekturer i boka

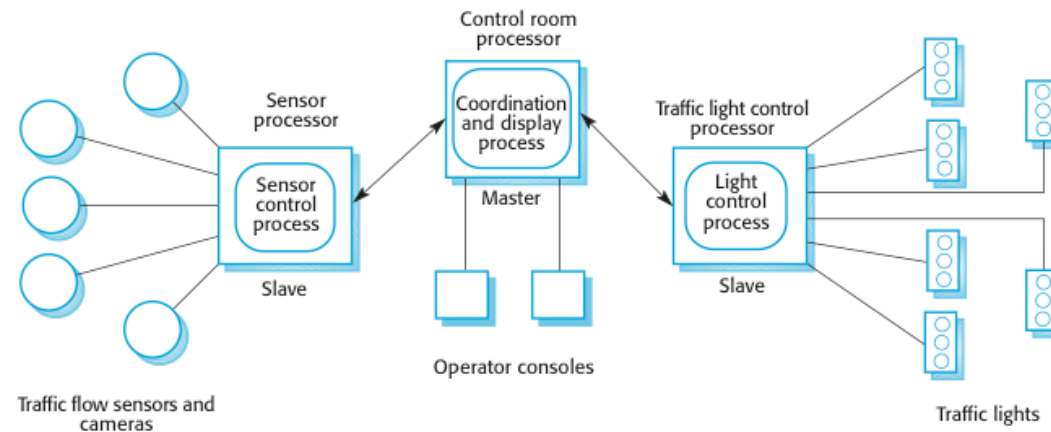
Hentet fra kapittelet om arkitektur i læreboka



Desentralisert P2P-arkitektur



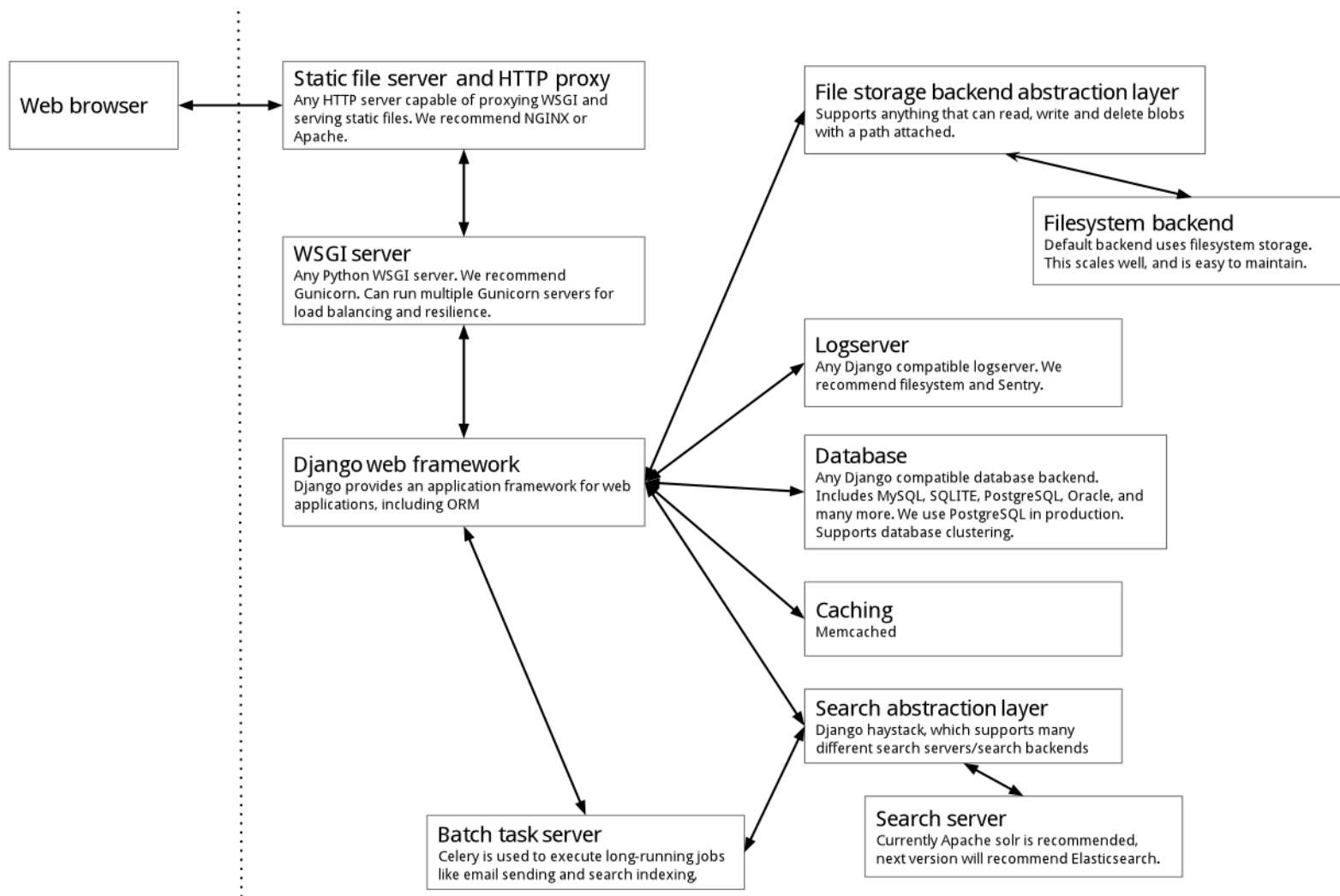
Tynn- og tykk-klient som arkitektur



Master-slave arkitektur eksemplifisert med et trafikklyssystem

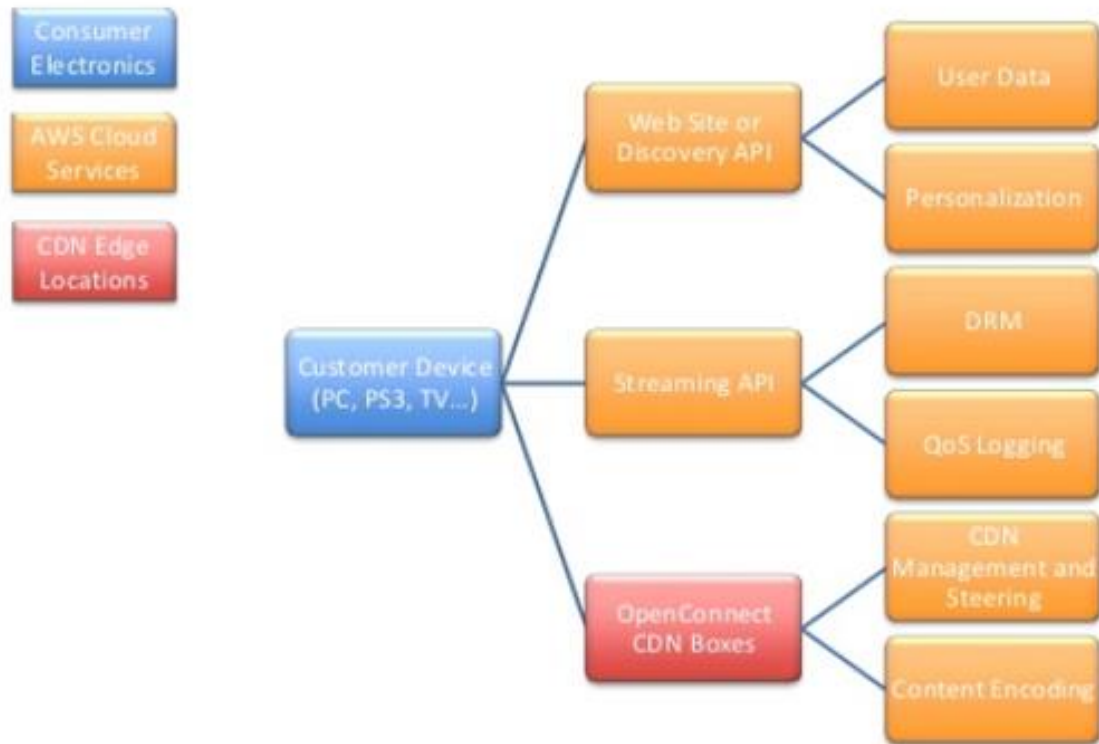
Eksempler fra virkeligheten:

Devilry



Netflix

How Netflix Works

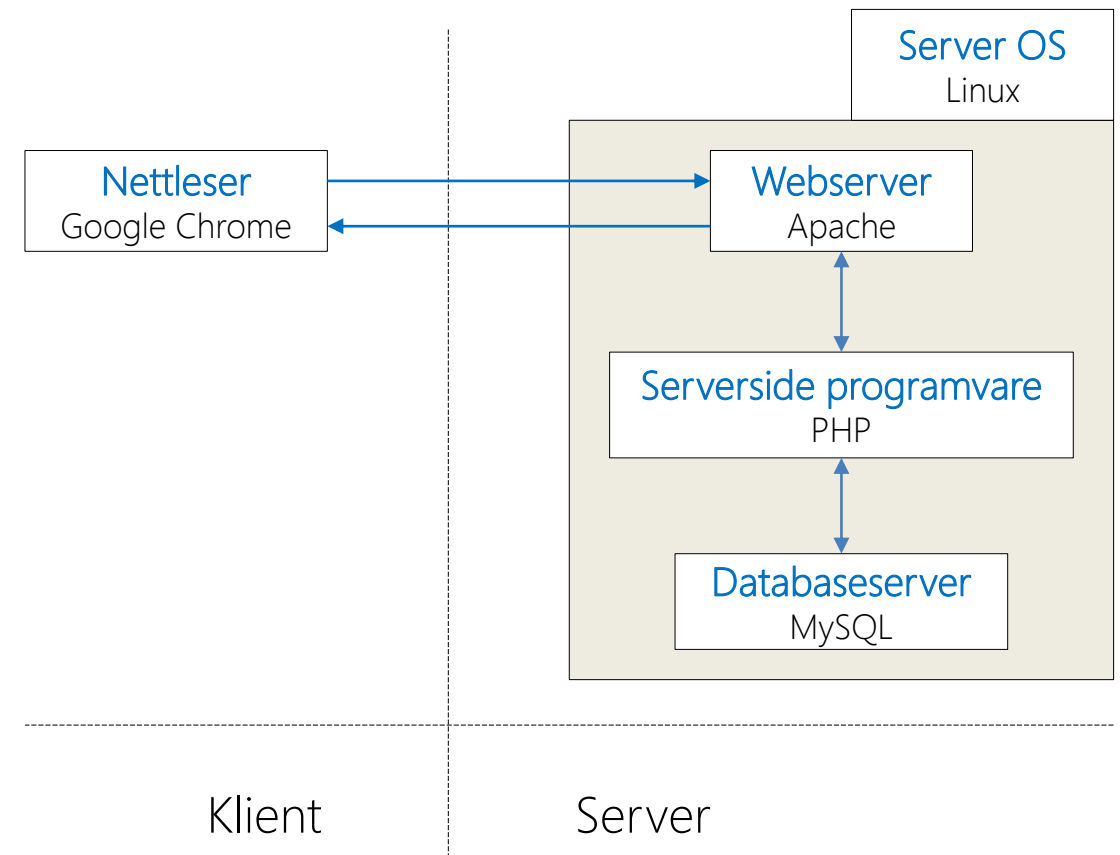


@adriano

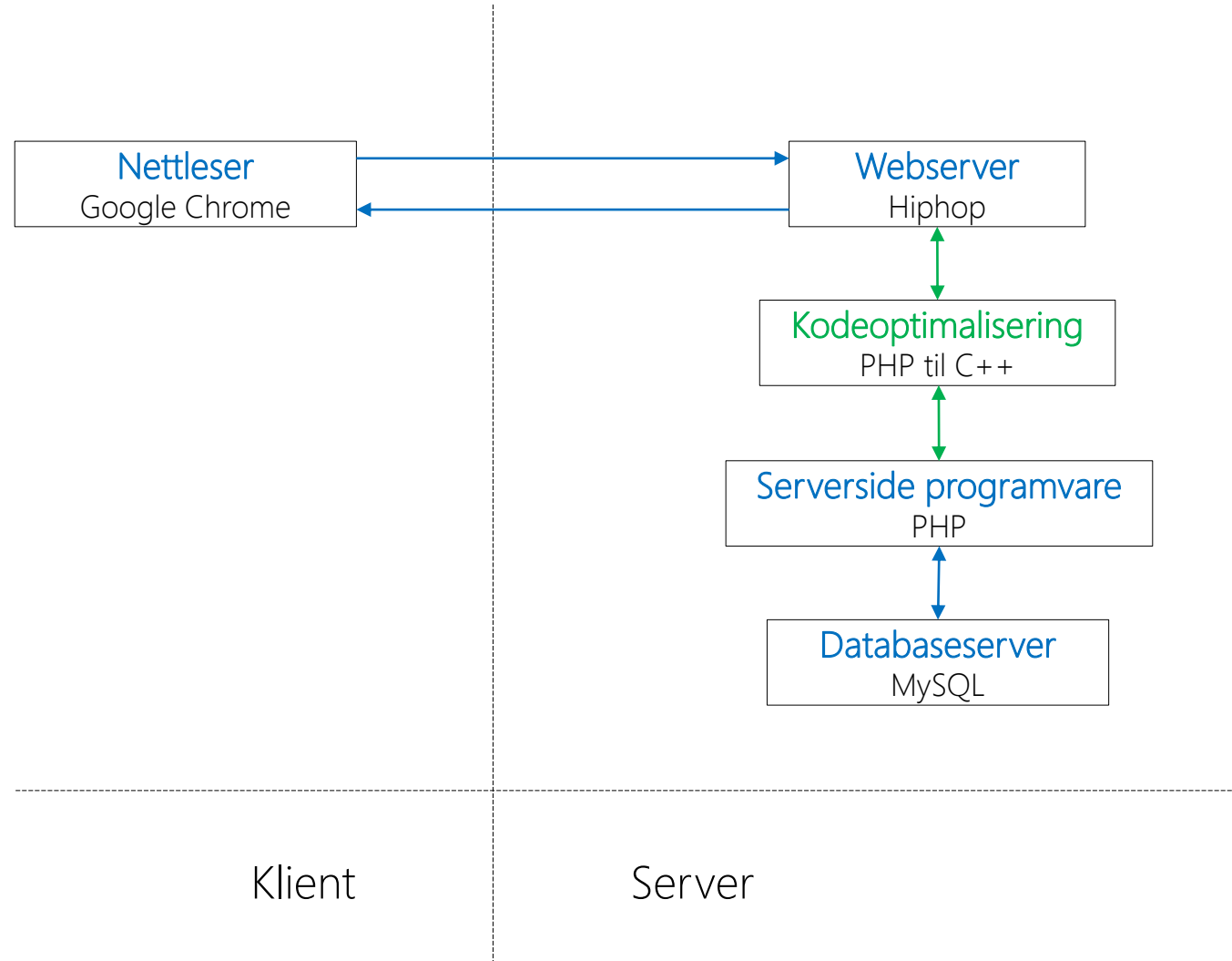


FBs arkitektur (1)

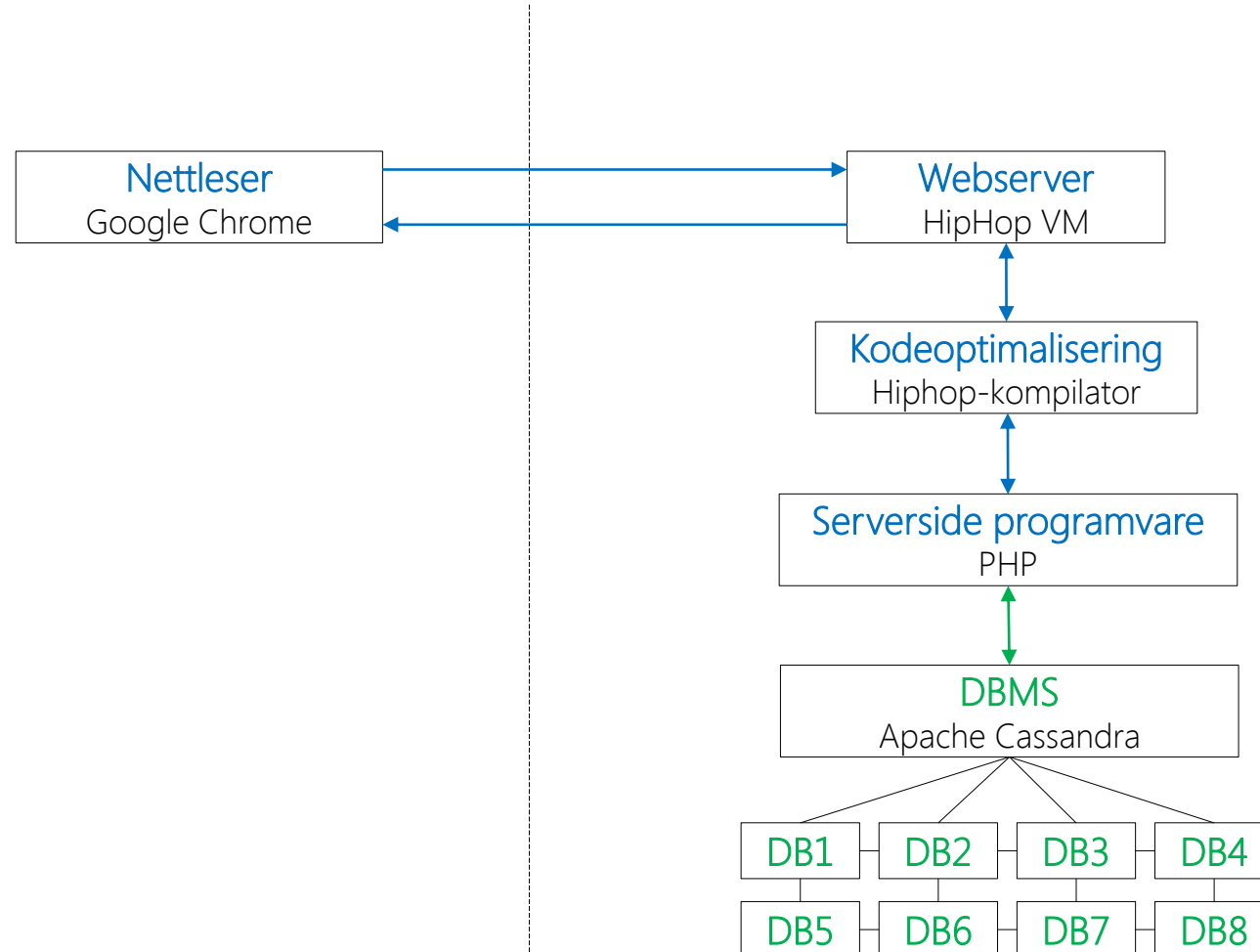
- Grunnprinsippet var LAMP (*Linux, Apache, MySQL, PHP/Perl/Python*)
- Utfordringer ved denne typen arkitektur:
- PHP er ikke nødvendigvis optimalt for store websider, skalerer ofte dårligere
- Skalering av en dynamisk database hvor nye relasjoner opprettes kontinuerlig er vanskelig
- MySQL anses som raskt og skalerbart, men krever ofte modifisering, f.eks. referanseintegritet, check constraints, ACID (via InnoDB)



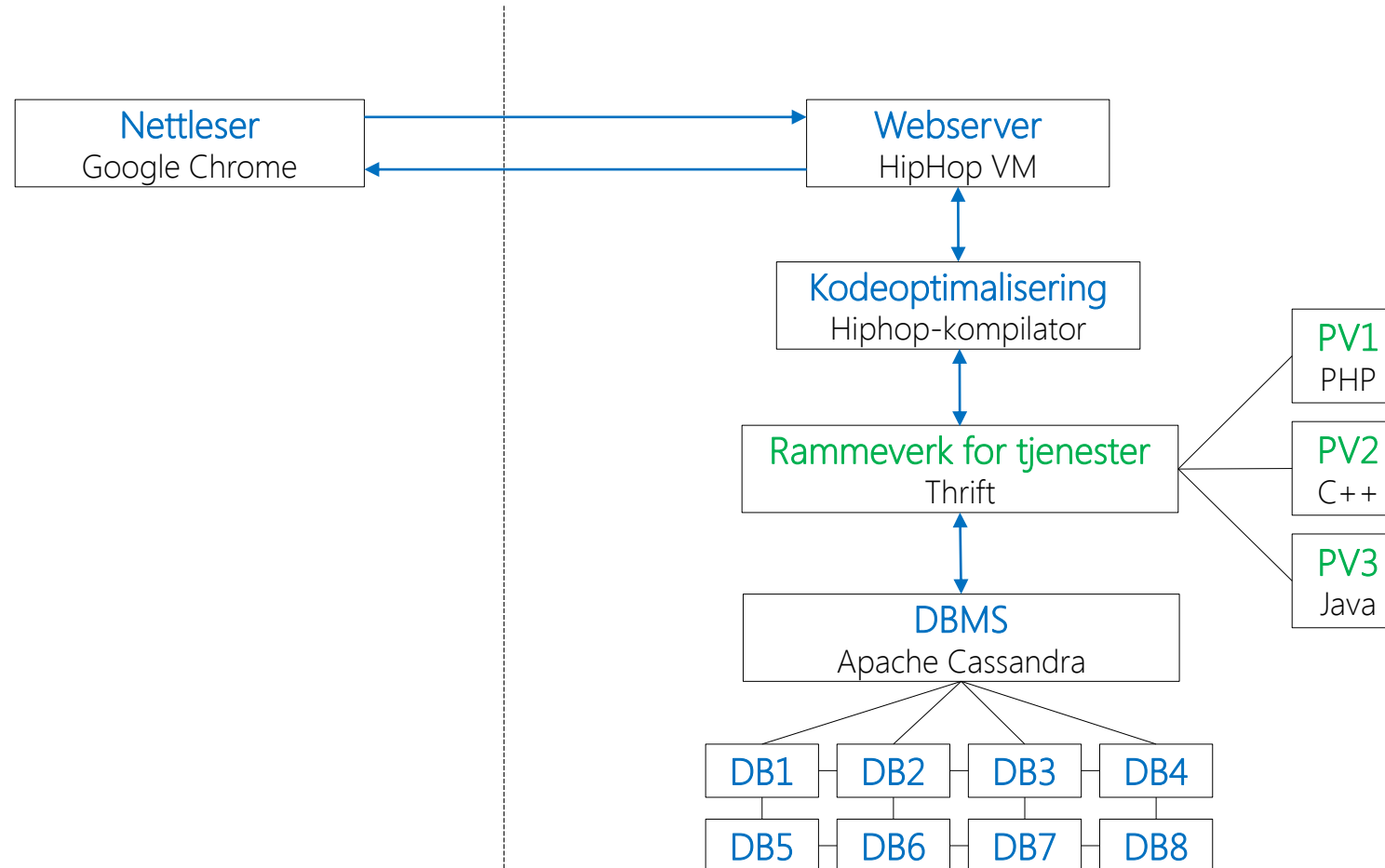
FBs arkitektur (2)



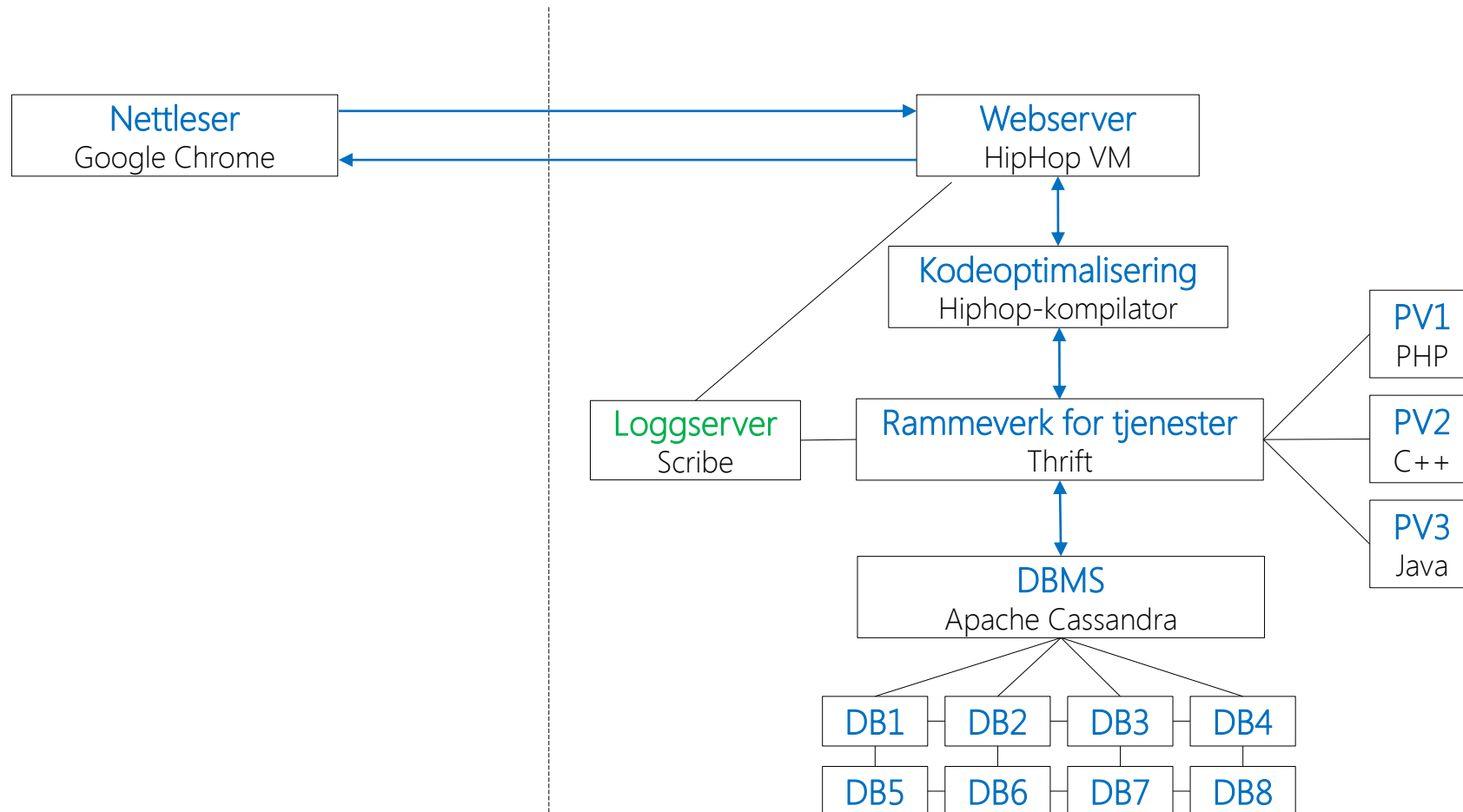
FBs arkitektur (3)



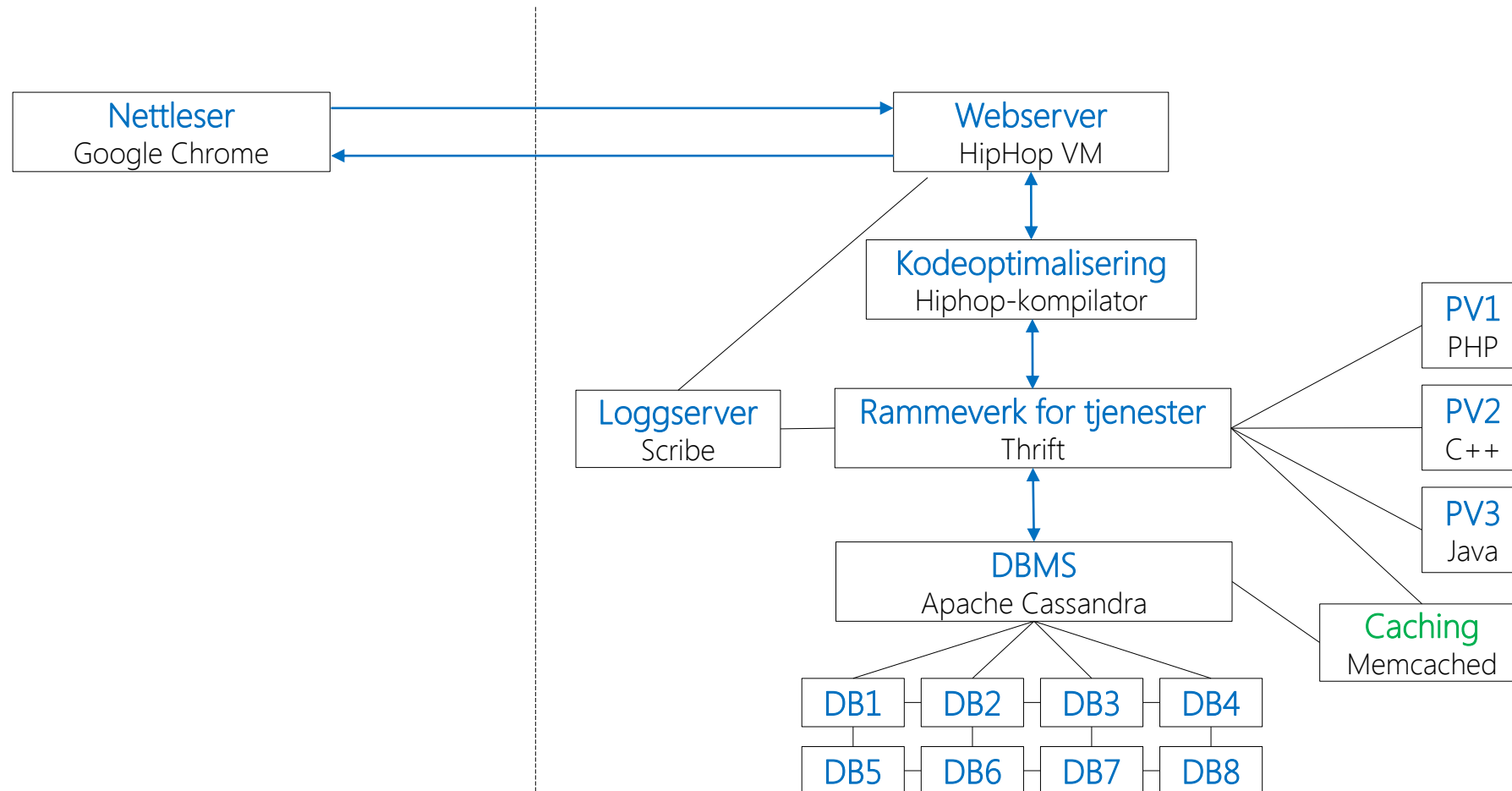
FBs arkitektur (4)



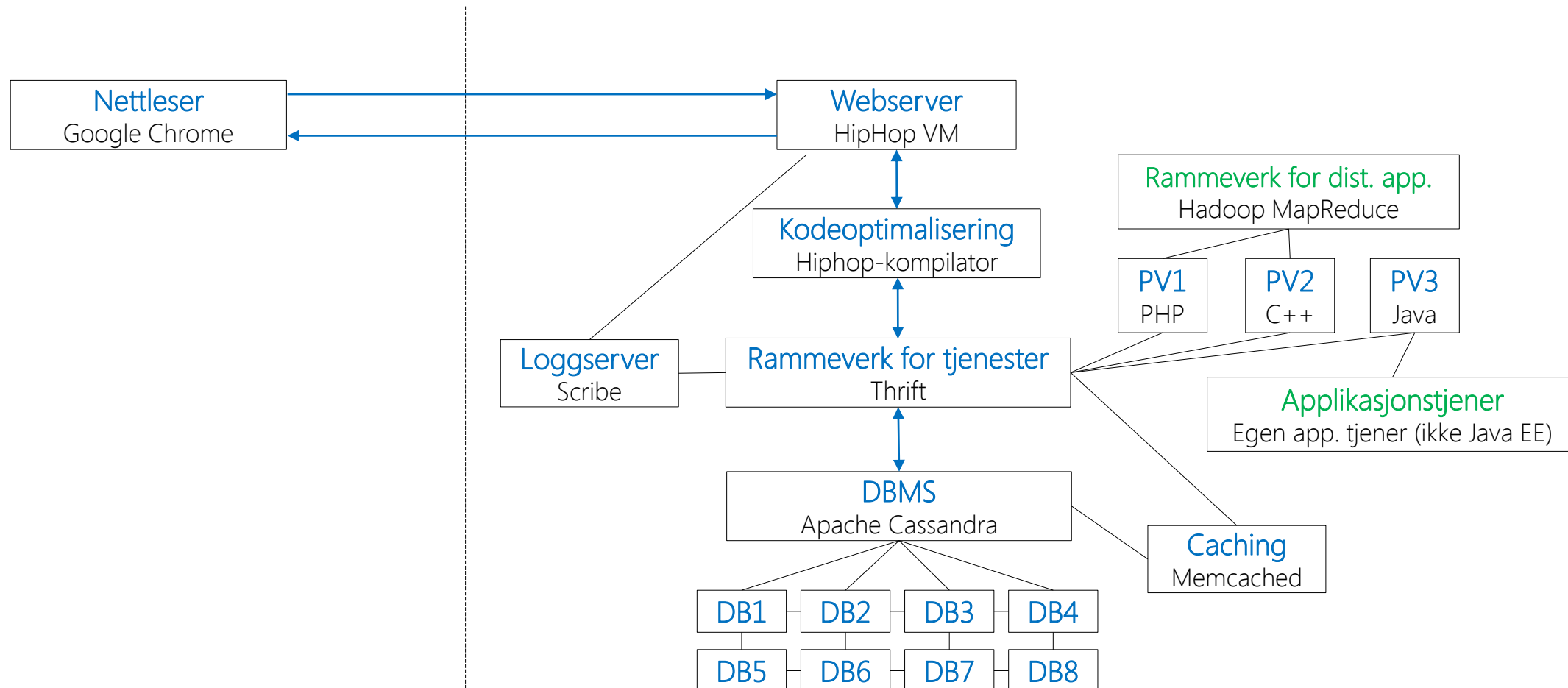
FBs arkitektur (5)



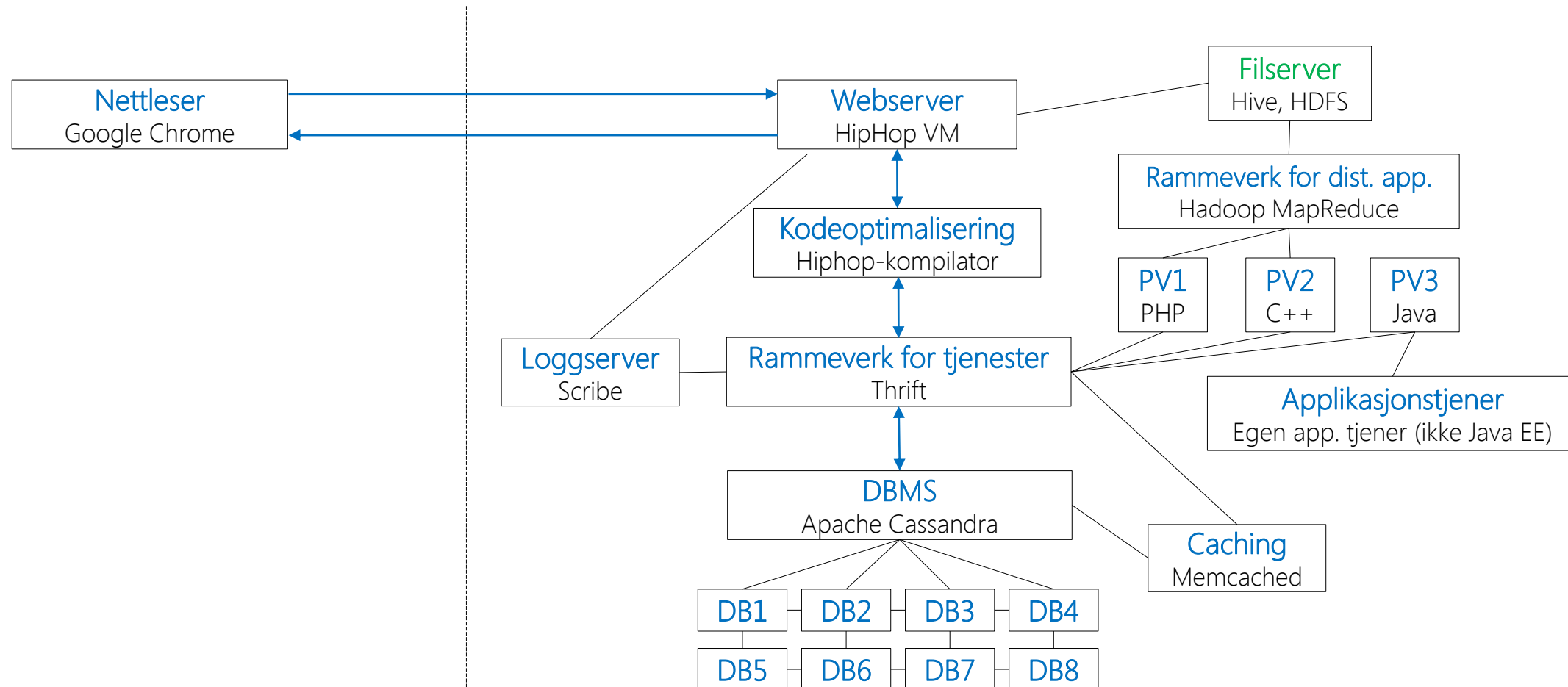
FBs arkitektur (6)



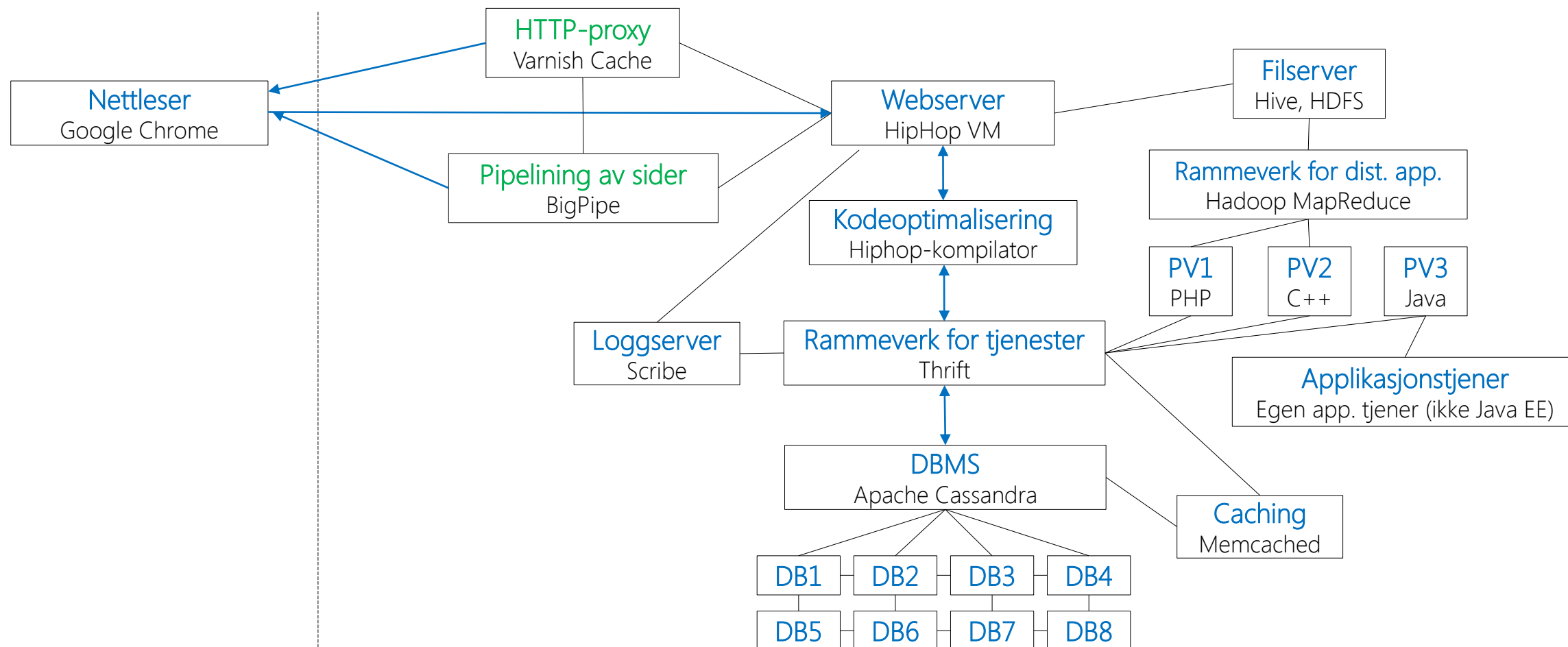
FBs arkitektur (7)



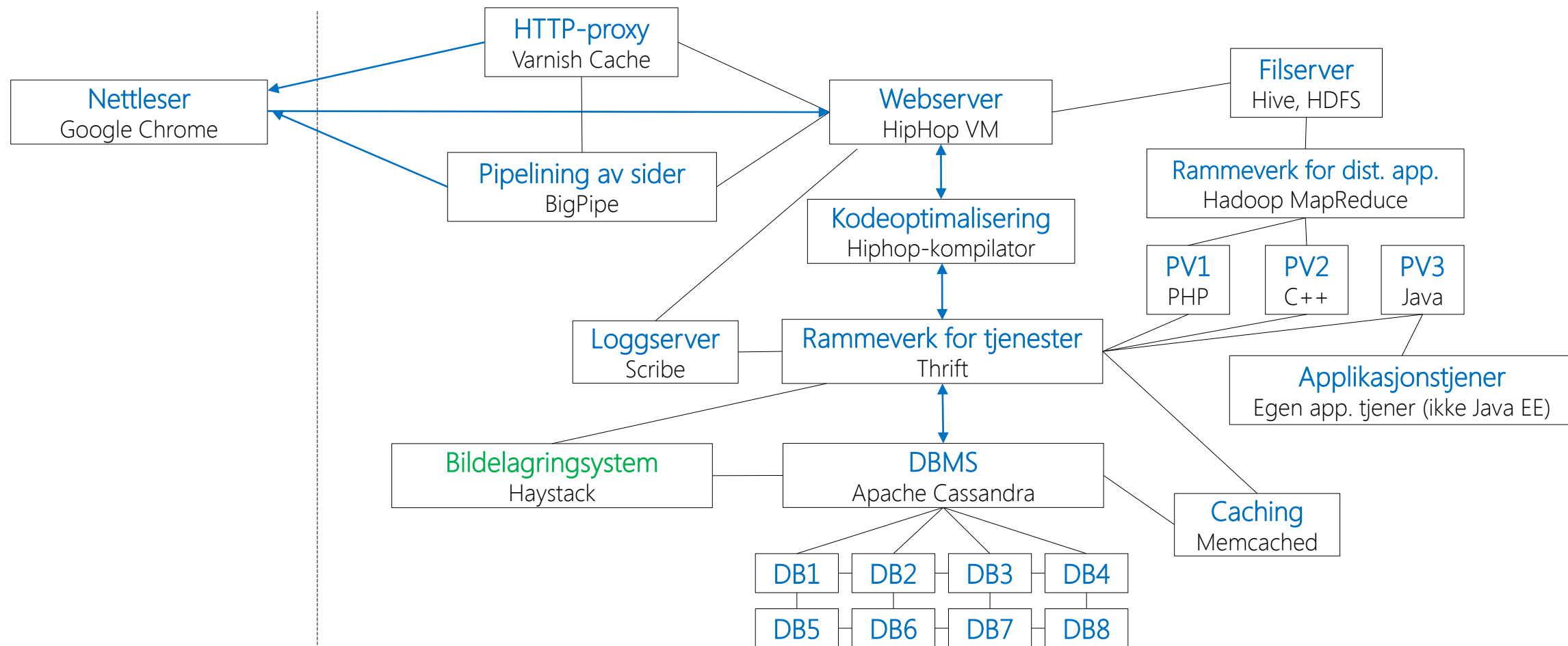
FBs arkitektur (8)



FBs arkitektur (9)



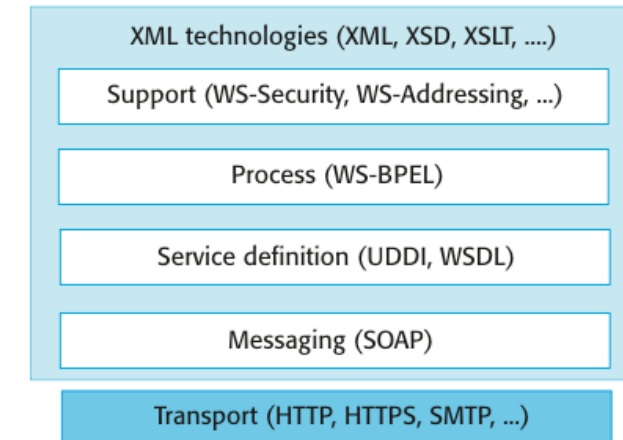
FBs arkitektur (10)



Service-oriented architecture (SOA)

Service-oriented architecture (SOA)

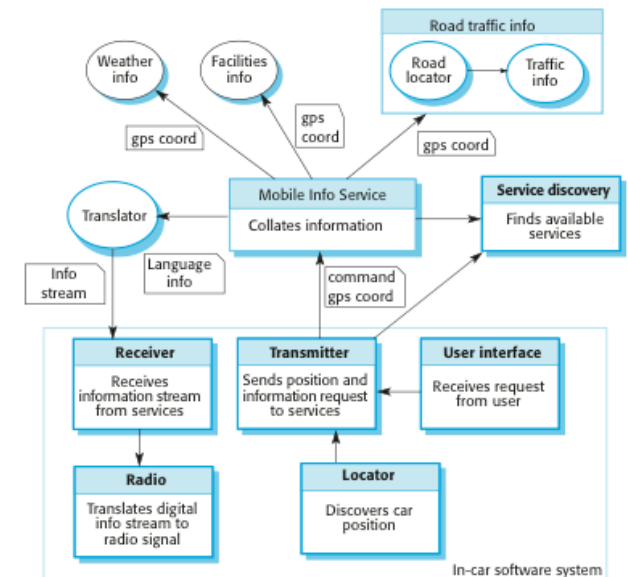
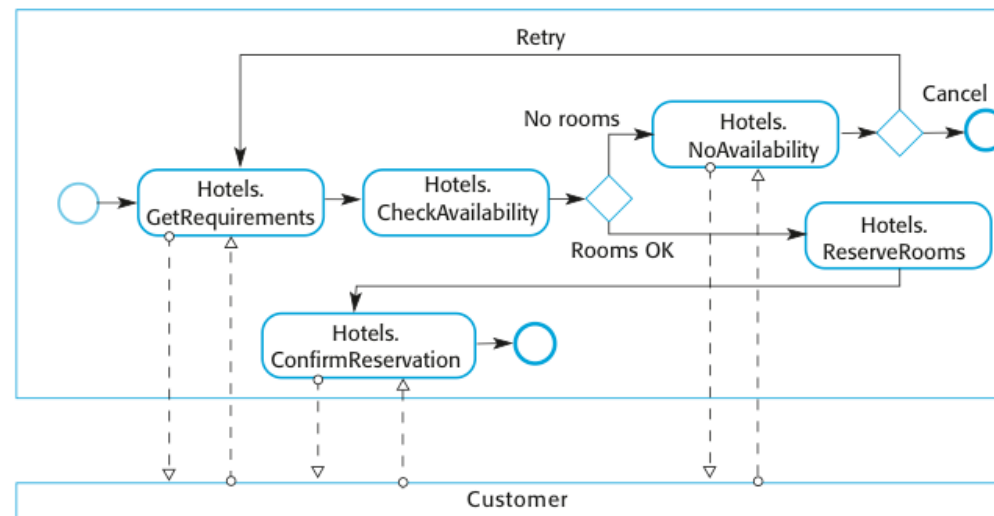
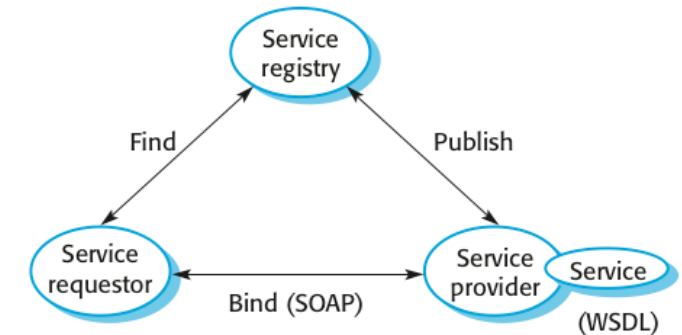
- Kalles for *tjenesteorientert arkitektur* på norsk
- Kostnadene ved å drive store systemer ble etter hvert veldig stor → SOA var resultatet
- Benytter seg av tjenester som er uavhengig av programvaren som bruker tjenesten
- Fokus på *loose coupling*, dvs. løse koblinger som er lette å erstatte, videreutvikle etc.
- Tjenestene er løst koblet og ikke assosierte med hverandre, og har som regel én funksjon
- Tjenestene bruker definerte protokoller og standarder for å sende beskjeder seg imellom
- Et av flere sentrale ledd i filosofien bak *SaaS* og *cloud computing*



Service-oriented architecture (SOA)

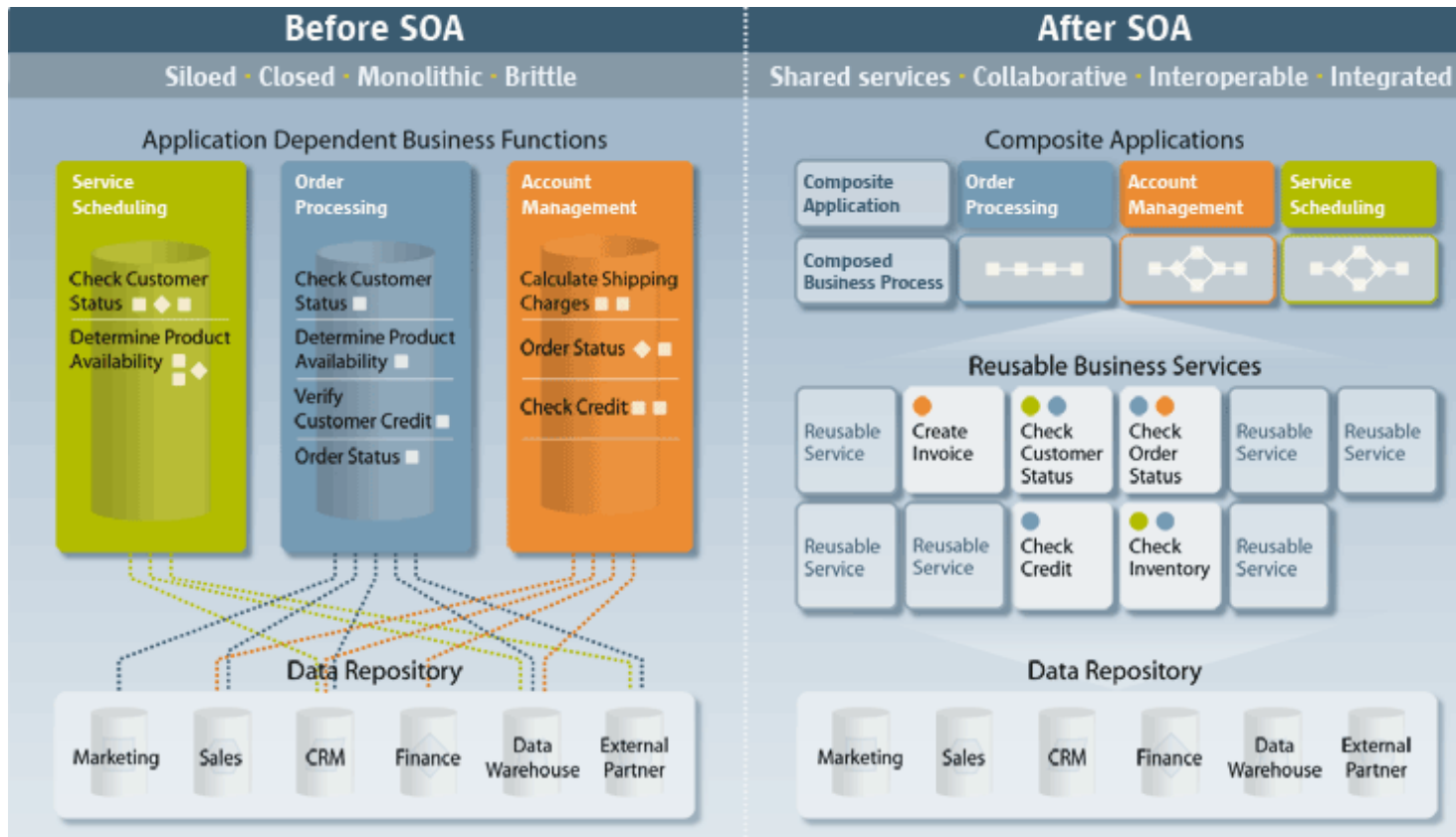
Må kunne redegjøre for fordeler med tjenesteorientert arkitektur og forklare hvorfor det løser mange av problemene ved tradisjonelle arkitekturer.

- Fordeler med SOA:
 - Tjenester kan tilføres lokalt eller outsources til eksterne tilbydere
 - Tjenestene er språkuavhengige
 - Kan kombinere tjenester fra ulike tilbydere i ett system
 - Kan bevare investeringer i eksisterende løsninger



Service-oriented architecture (SOA)

- Eksempel på integrasjon av SOA



(http://cosmosys.in/ig_soa_before.gif)

Service-oriented architecture (SOA)

3-lags arkitektur	SOA
Homogent	Heterogent
Språkavhengig	Språkuavhengig
Sentralisert applikasjonslag	Massive distribuerte tjenester
Kodesentrert applikasjon	Fleksibel sammensatt applikasjon
Spørring/svar-drevet	Spørring/svar, publish-subscribe, hendelser
HTML-sider	AJAX Rich Internet Application (RIA)

Takk for meg!

- Tilbakemeldinger, kritikk, feil i lysark og spørsmål mottas med takk:

joshi@ifi.uio.no

Appendix 1: Systemarkitektur de siste ti år

- Inkorporering av arkitektoniske notasjoner og begreper i mainstream *designspråk* (f.eks. UML2), og *verktøy* (f.eks. Rose)
- *Metoder* basert på arkitektonisk design (f.eks. MDA)
- Enkelte arkitektoniske *analyseverktøy* (f.eks. AcmeStudio)
- Arkitektoniske *standards* for enterprise-systemer (f.eks. RM-ODP, TOGAF)

(J.P. Sousa, 2011)

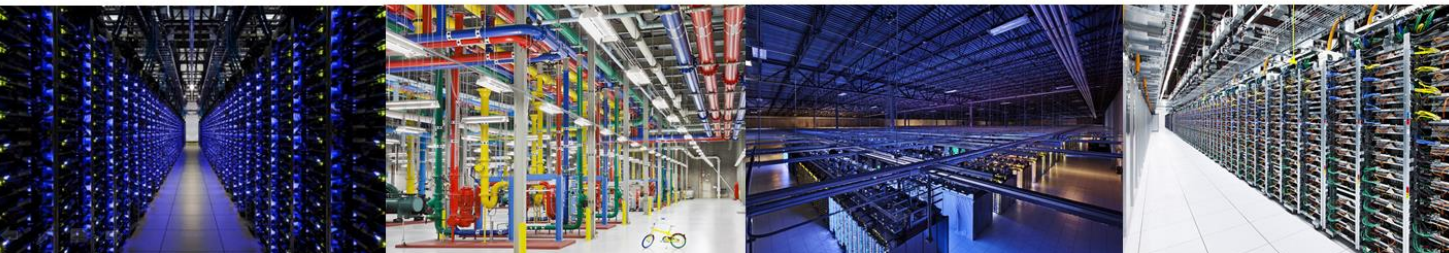
Appendix 2: YouTube

(Mine lysark fra en annen forelesning ved Ifi, 1. november 2012)

YouTube

Mye vil ha mer?

- *Hvert minutt lastes det opp 60 timer med video*
- *Hvert sekund lastes det opp 1 time med video*
- *Mer video er lastet opp på YouTube i løpet av én måned enn de tre største produksjonsselskapene har klart å produsere de siste 60 år*
- *10 % av all internettrafikken som genereres stammer fra YouTube*
- *Tredje mest besøkte nettsiden i verden (hvem er mer populær?)*



Appendix 3: Big data

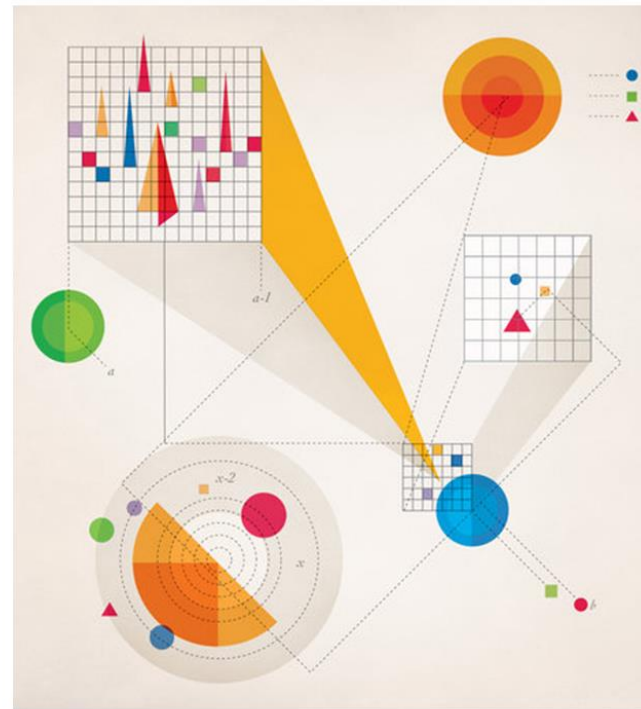
(Mine lysark fra en annen forelesning ved Ifi, 1. november 2012)

12

Big data

Datamengder ute av kontroll?

- Tradisjonelt: gendata, værdata, børldata etc.
- **Walmart:** 1 million kundetransaksjoner hver time (277 i sekundet) → utgjør 2,5 petabyte
- **Facebook:** bildedatabasen har over 40 milliarder bilder (som skal kunne hentes frem på sekundet)
- Estimerer anslår at volumet på all businessdataen worldwide dobles hvert 14-15 måned
- Det genereres 1 exabyte med data hver eneste dag
- **Datamining** → forstå sammenhenger, korrelasjoner og utlede mønstre



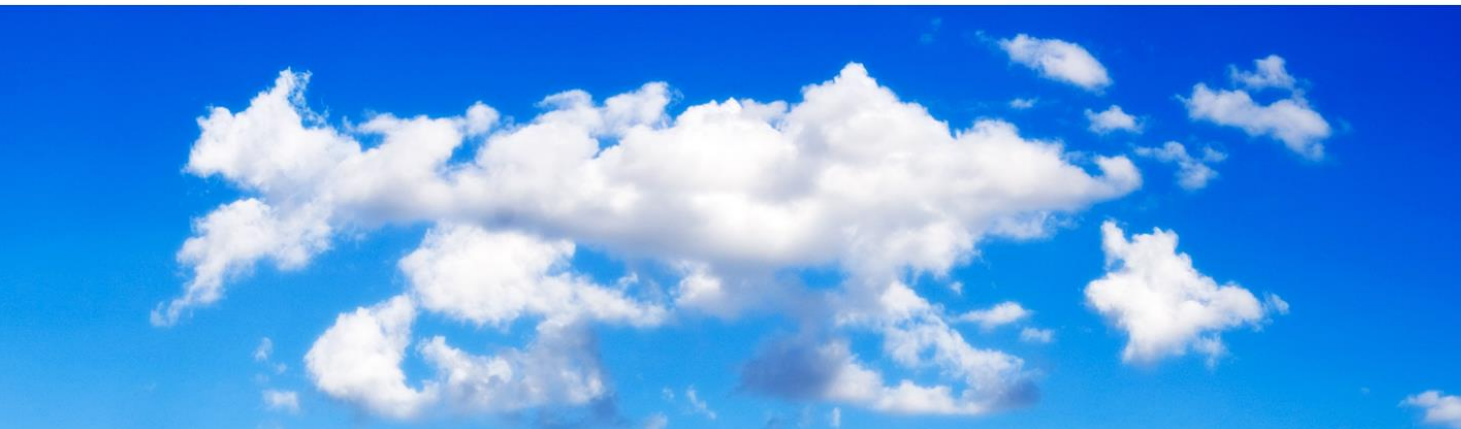
Appendix 4: Cloud computing

(Mine lysark fra en annen forelesning ved Ifi, 1. november 2012)

Cloud computing

La oss dytte alt i opp i nettskyen

- *Ekstern datalagring*
- *Økt tilgjengelighet*
- *Dynamisk skalering gir kostnadseffektivitet og fleksibilitet*
- *Sikkerhetsspørsmål → hvor lagres dataen og hvem har tilgang?*



Appendix 5: Videre lesing

- Hvilken rolle har en arkitekt?
http://www.bredemeyer.com/pdf_files/role.pdf
- Eksempler og historier omkring arkitektur fra den virkelige verden
<http://realworldsa.blogspot.com>
- Yahoo sitt utviklingsmiljø og deres arkitektoniske filosofi (video)
<http://effectivesoftwaredesign.com/2011/11/16/yahoo-cocktails-when-the-client-is-in-the-cloud>
- Microsoft sin teknikk for implementering av sikkerhet og feilhåndtering i arkitektoniske modeller
<http://msdn.microsoft.com/en-us/library/ee658084.aspx>

Appendix 6: Marsmallow challenge

- Se TED-talk her: http://marshmallowchallenge.com/TED_Talk.html

