TNM094 – Medietekniskt kandidatprojekt

Systemarkitektur

Systemarkitektur

- Bygger upp systemets generella form
  - vilka datorer, program, processer
- Delar upp systemets kodbas
  - vilka kodmoduler, ramverk, APIer

Arkitekturell modellering

- För att...
  - ...förstå systemet
  - ...skapa en ritning för konstruktion
  - ...analysera beroenden
  - ...hitta återanvändning av delar
  - ...resonera kring framtida förändringar
  - ...stödja ledningsbeslut och riskhantering

[Pfleeger]

Vyer

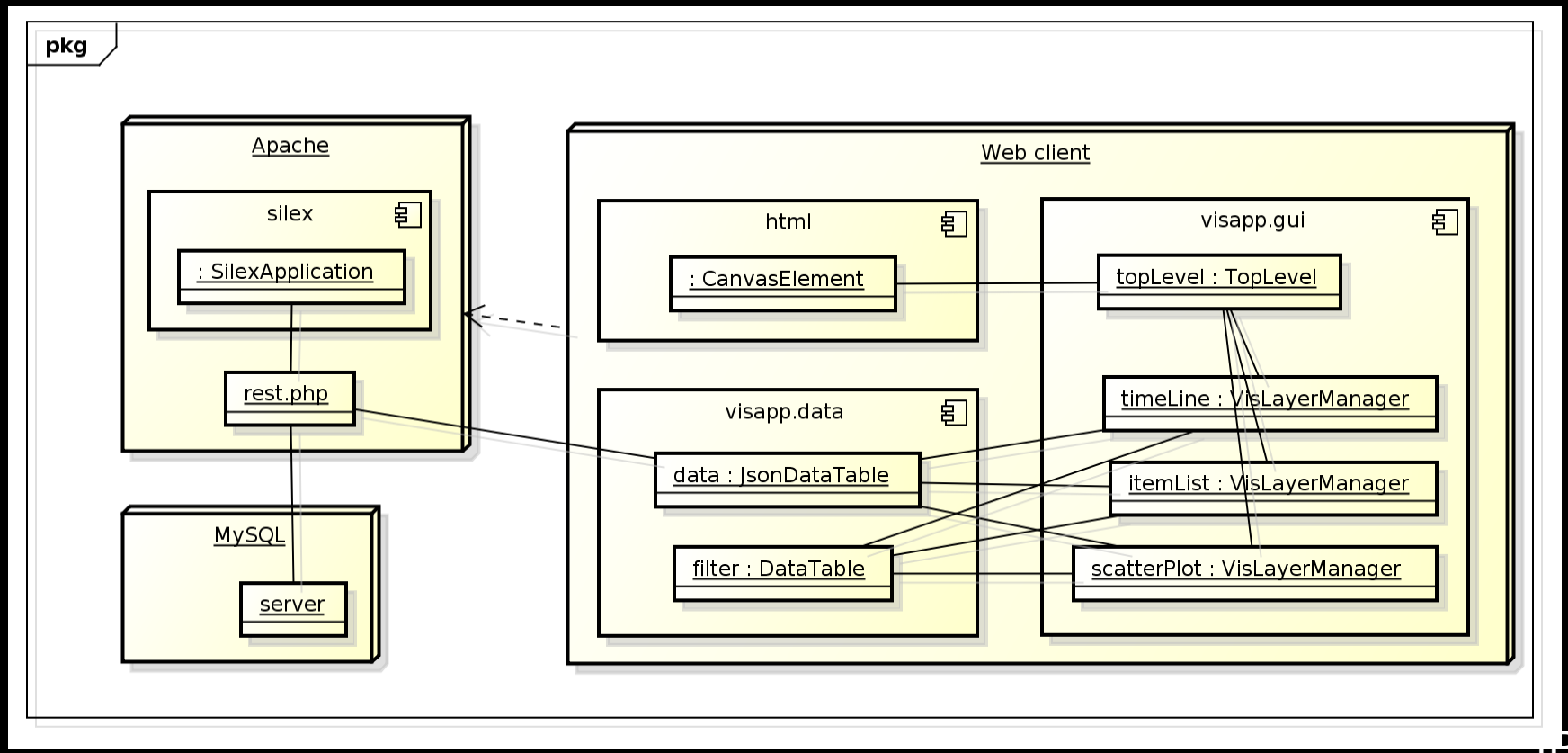
- Ett system, flera modeller, olika vyer
- Varje vy visar en specifik aspekt av systemet
  - kodstruktur (paket och moduler)
  - körnings-struktur (server, klient, nätverk, shm, etc)

  etc.

[Pfleeger]

Körnings-struktur

- Hur ser systemet ut internt under körning
  - Flera vyer om det kan se olika ut
  - Instanser, processer, datastrukturer, anropsvägar



[Pfleeger]

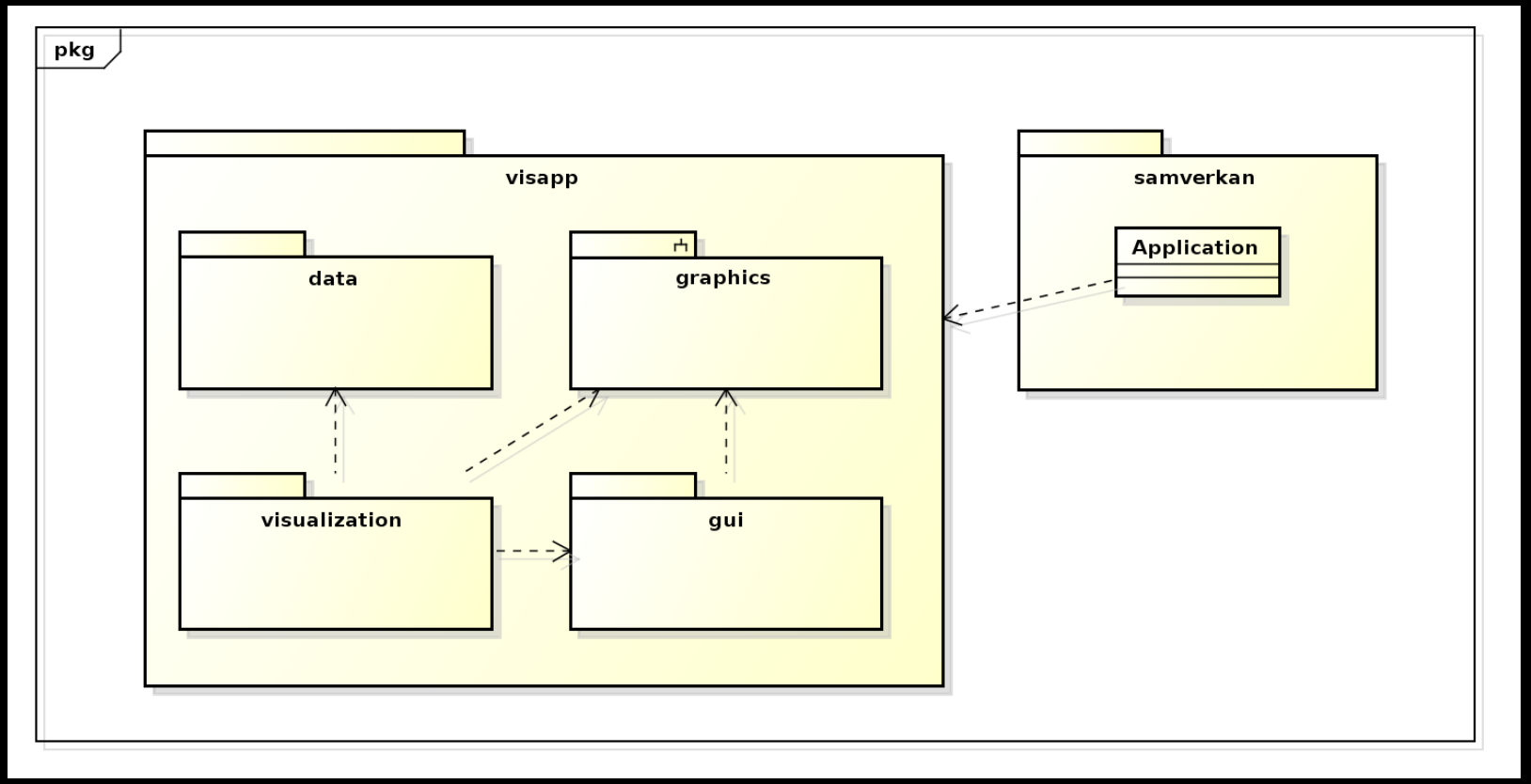
Viktiga element

- Vad finns där?
  - Noder
    - datorer, servrar, processer, program, etc
  - Gränssnitt
    - process-anrop, abstraktion, nätverkskommunikation, etc
  - Programvaruenheter
    - oftast moduler och submoduler
- Vad finns inte med?
    - implementations-specifika algoritmer
    - variabler, klasser, objekt, funktioner

[Pfleeger]

Kodstruktur

- Vilka moduler har systemet (systemets kodbas)
  - Hur förhåller sig dessa till varandra
  - Paket, bibliotek, filer, enheter, komponenter



[Pfleeger]

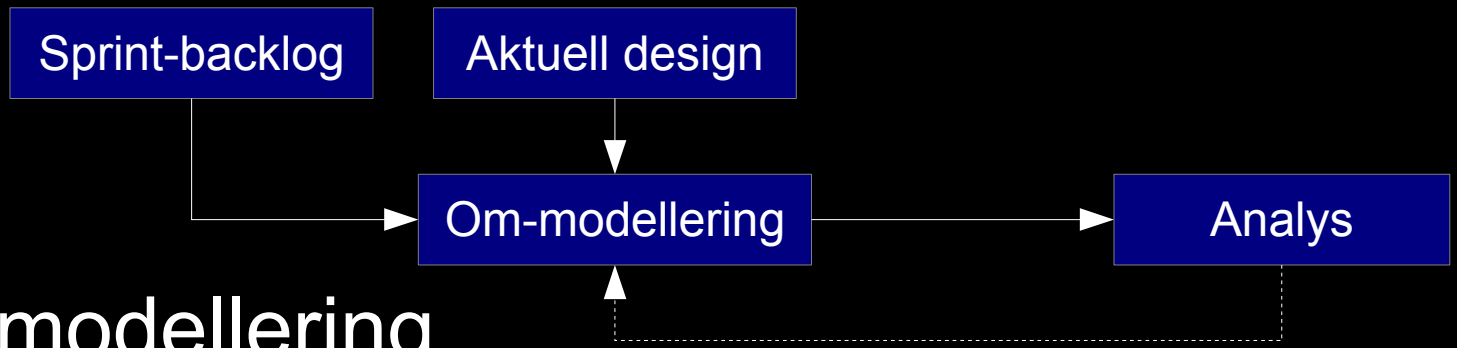
Några viktiga termer

programvaruenhet: del av ett system som kan relateras till som en enhet, t ex utifrån dess roll. Kan rent praktiskt vara en klass, modul eller någonting annat.

gränssnitt: del av en programvaruenhet som syns och används av andra enheter, ibland även för ett specifikt syfte.

[Pfleeger]

Design-process för Agil utveckling

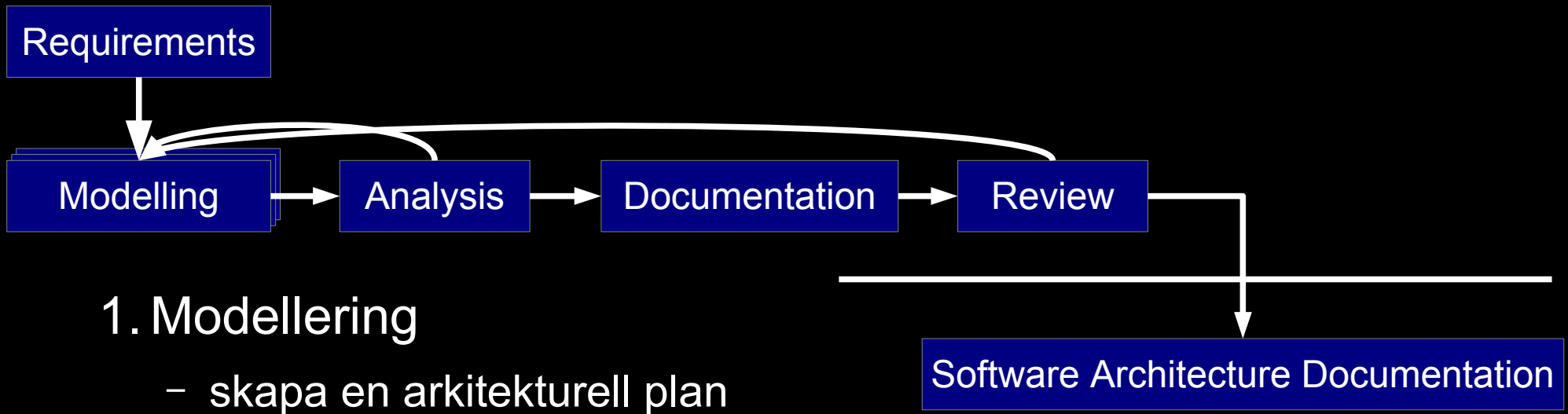


- Om-modellering
  - Analysera nya krav i sprint-backloggen
  - Förändra den nuvarande designen för de nya kraven
  - Rita och diskutera – spara alla möjliga alternativ
- Analys
  - Jämför de nya alternativen utifrån olika kvalitetsattribut
  - Välj det bästa alternativet
  - (Dokumentera logisk grund vid behov)

Systemarkitektur för Agil utveckling

- Bara precis tillräckligt för tillfället
  - tillräckligt för nuvarande funktionalitet och nästa sprint
  - typiskt väldigt begränsad framförhållning
  - (RDUF – Rough Design Up Front)
- Gör om arkitekturen vid behov
  - inför varje sprint, uppdatera arkitekturen

Design-process för dokumentation



1. Modellering
   - skapa en arkitekturell plan
   - fatta beslut på system-nivå
2. Analys
   - bedöm hur väl modellen passar systemkraven
3. Dokumentation
   - dokumentera modeller och vyer
   - hitta och dokumentera beroenden
4. Granskning
   - formell granskning och verifikation

[Pfleeger]

Modellering av systemarkitektur

- Rutor, text och pilar
  - Rita fritt och enkelt
- UML
  - Kodstruktur: Klassdiagram
  - Körnings-struktur: Komponentdiagram, Deployment-diagram
  - Steg-för-steg
    Use case-diagram → Aktivitetsdiagram
    → Komponentdiagram (→ Klass d.) (Five-step-UML)

[Pfleeger]

Källor för designval

- Referensmodeller
  - generella eller domänspecifika
- Andra liknande system
- Erfarenhet
- Konventioner och idiom
- Design Patterns
- (Emergent design)

[Pfleeger]

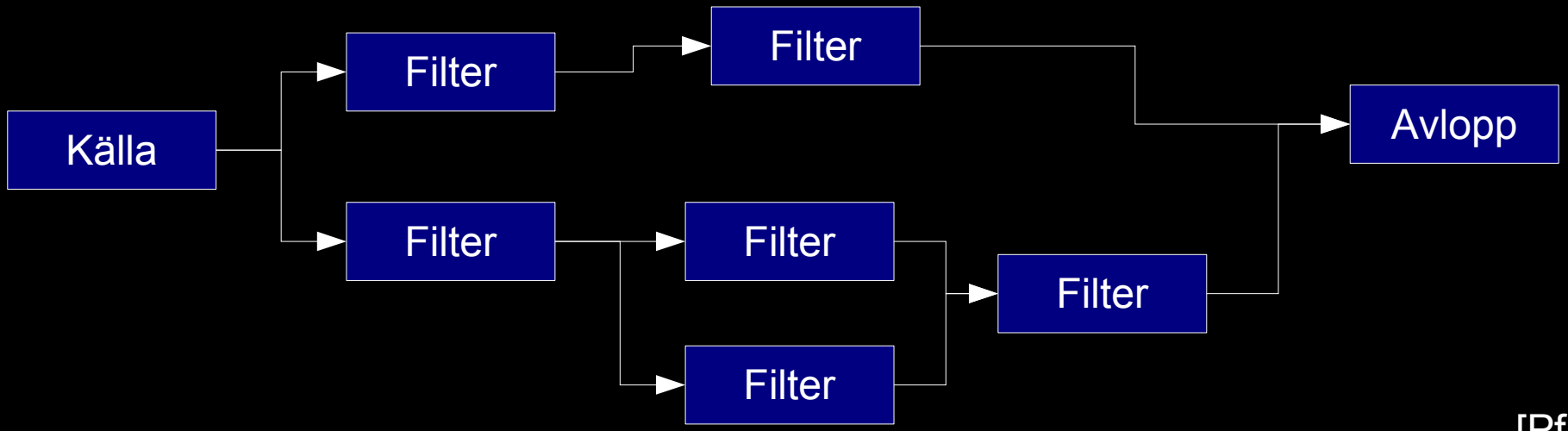
Stilar och strategier

- Hög-nivå strukturer och mönster
  - inkompletta, generella mallar
  - välkända egenskaper
- Kombinera stilar till ett komplett system
  - i olika delar av systemet
  - på olika nivåer i systemet

[Pfleeger]

Pipe-and-Filter

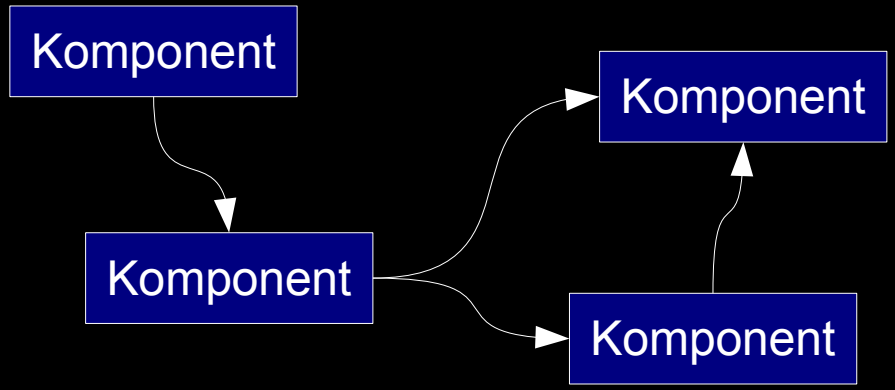
- Principer
  - Standardiserat data-format och filter-principer
  - Stödjer dynamisk och statisk konfiguration
- Egenskaper
  - Återanvändbarhet, anpassning, beteendejustering
  - Mindre flexibilitet i data-format och filter



[Pfleeger]

Publish-Subscribe

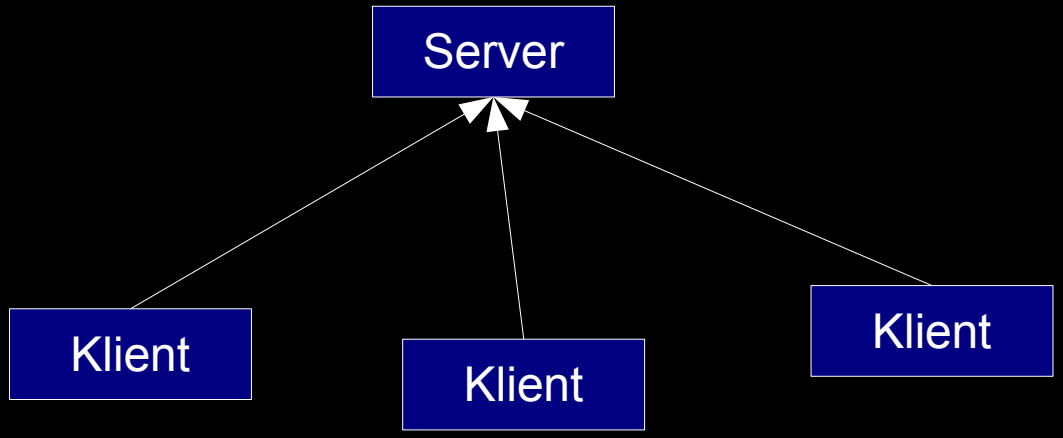
- Principer
  - en komponent publicerar data andra tar emot (publicist → prenumerant)
- Egenskaper
  - utbyggbart och främjar återanvändning av moduler
  - skicka och synkronisera data kan vara dyrt



[Pfleeger]

Klient-Server

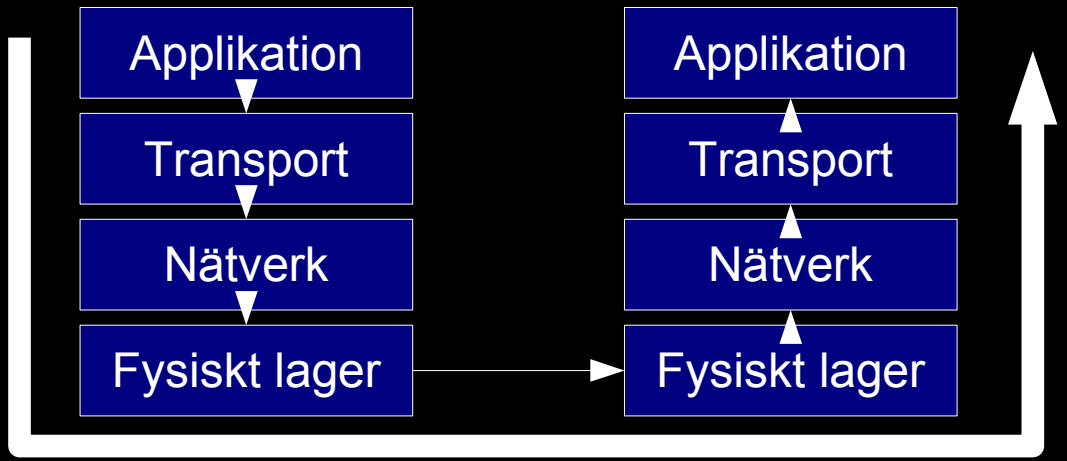
- Principer
  - Förfrågan → svar
  - Asymmetrisk förhållande, stjärn-arkitektur
- Egenskaper
  - Centraliserad kontroll
  - Potentiell flaskhals och enskild svag punkt (single-point-of-failure – SPOF)



[Pfleeger]

Layering

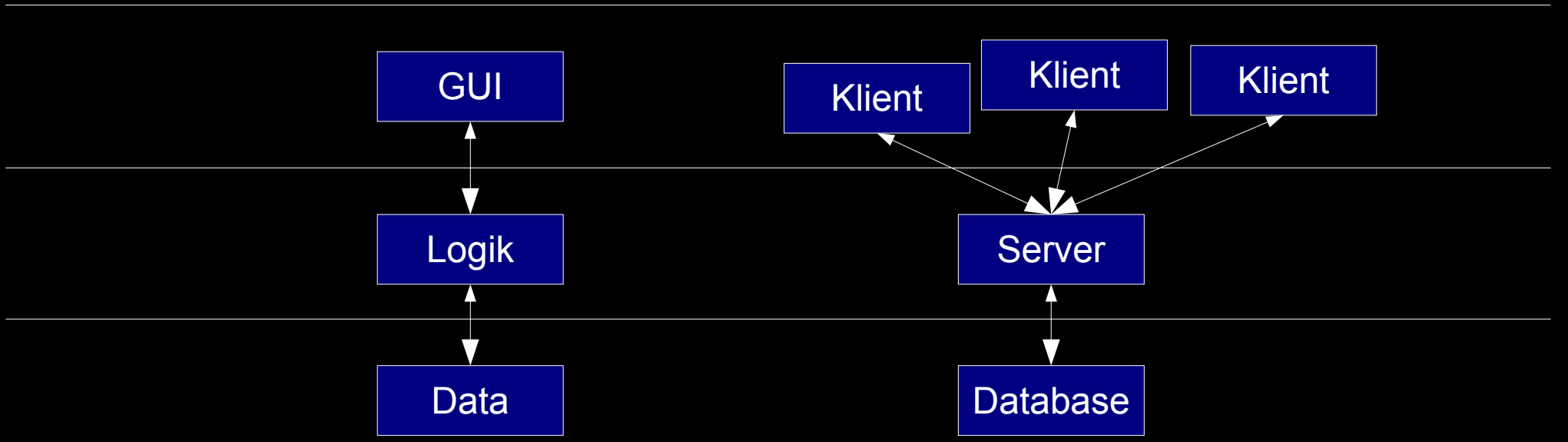
- Principer
  - hierarkisk struktur baserad på abstraktion
  - tillåter att anrop hoppar över ett lager
- Egenskaper
  - förenklat gränssnitt och struktur
  - förenklar återanvändning av moduler
  - möjligen viss negativ effekt på prestandan



[Pfleeger]

Three-/Multi-tier System

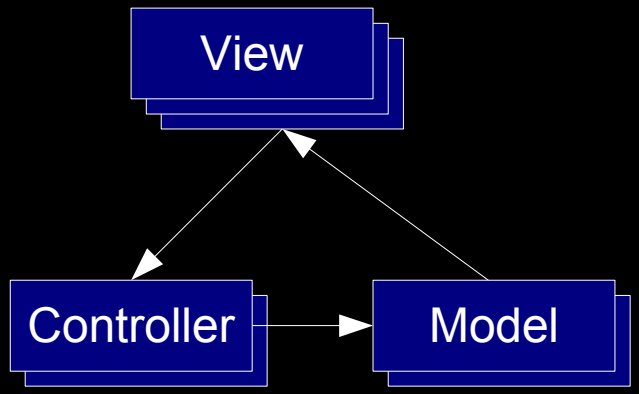
- Principer
  - strikt uppdelat ansvar, ”kommunicera genom”
- Egenskaper
  - välkända och separerade ansvar och funktioner



[Pfleeger]

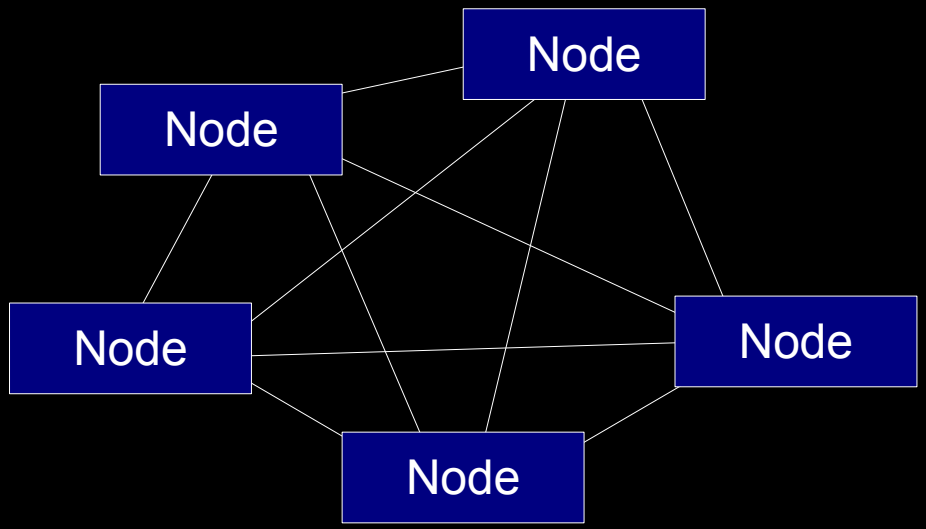
Model-View-Controller

- Principer
  - uppdelning av ansvar, riktad kommunikation
- Egenskaper
  - renare programkod, enklare att hitta fel
  - enkelt att lägga till flera vyer av en datamodell



Peer-to-Peer

- Principer
  - alla komponenter kommunicerar direkt
- Egenskaper
  - låg fördröjning, ingen flaskhals
  - svårt att kontrollera och synkronisera
  - svårt att förändra och avlusa



[Pfleeger]

Kvalitetsattribut

- Hur vet vi om en arkitektur är bra?
- Viktiga aspekter hos en design
  - Hög sammanhållning, låg koppling
  - Modifierbarhet
  - Prestanda
  - Säkerhet
  - Pålitlighet
  - Robusthet
  - Användbarhet
  - Affärsmål

[Pfleeger]

Sammanhållning och koppling

sammahållning (cohesion): hur väl en enhets olika delar passar ihop.

koppling (coupling): hur starkt beroende två enheter har av varandra, d v s hur de påverkas av en eventuell förändring i den andra.

[Pfleeger]

Modifierbarhet

- Hur lätt är det att senare justera en design?
- Designa för framtida förändring
  - Kontrollera direkt och indirekt påverkade enheter
  - Minimera mängden direkt påverkade enheter
    - förutse förväntade förändringar
    - sträva efter sammanhållning inom enheter
    - sträva efter generella gränssnitt mellan enheter
  - Minimera mängden indirekt påverkade enheter
    - håll nere koppling (beroendet) – använd gränssnitt
    - behåll gamla gränssnitt om möjligt och skapa nya vid behov

[Pfleeger]

Prestanda

- Hur bra prestanda kommer systemet att få?
  - Effektiv resurs-allokering
  - Svara på anrop så tidigt som möjligt
  - Kasta bort data så tidigt som möjligt
  - Kopiera stora mängder data sällan
- Krav på prestanda komplicerar ofta arkitekturen

[Pfleeger]

Säkerhet

- Två sorters säkerhet
- Elasticitet
  - återhämta sig snabbt från angrepp
  - isolera systemdelar för att minimera effekten av intrång
  - använd strategier för att snabbt återfå funktionalitet
- Immunitet
  - påverkas inte av attacker
  - innefatta säkerhet i er design
    (Inte säkerhet genom fördunkling – security by obscurity)

[Pfleeger]

Pålitlighet / robusthet

- Två sorters påligtlighet
- Pålitlighet
  - fungerar korrekt under antagna förhållanden
- Robusthet
  - fungerar korrekt även vid felaktig input
  - fungerar korrekt vid oväntad belastning

[Pfleeger]

Användbarhet

- Att tänka på
  - användargränssnitt förändras ofta
    → separera och kapsla in
  - vissa funktioner (t ex undo) kräver stöd i arkitekturen

[Pfleeger]

Affärsmål

- Mål
  - kostnad att utveckla
  - tid till marknad
  - stöd för olika plattformar
  - kvalitets- och funktionsnivå
- Beslut
  - köp eller implementera
  - utveckling eller underhåll
  - ny eller beprövad teknik