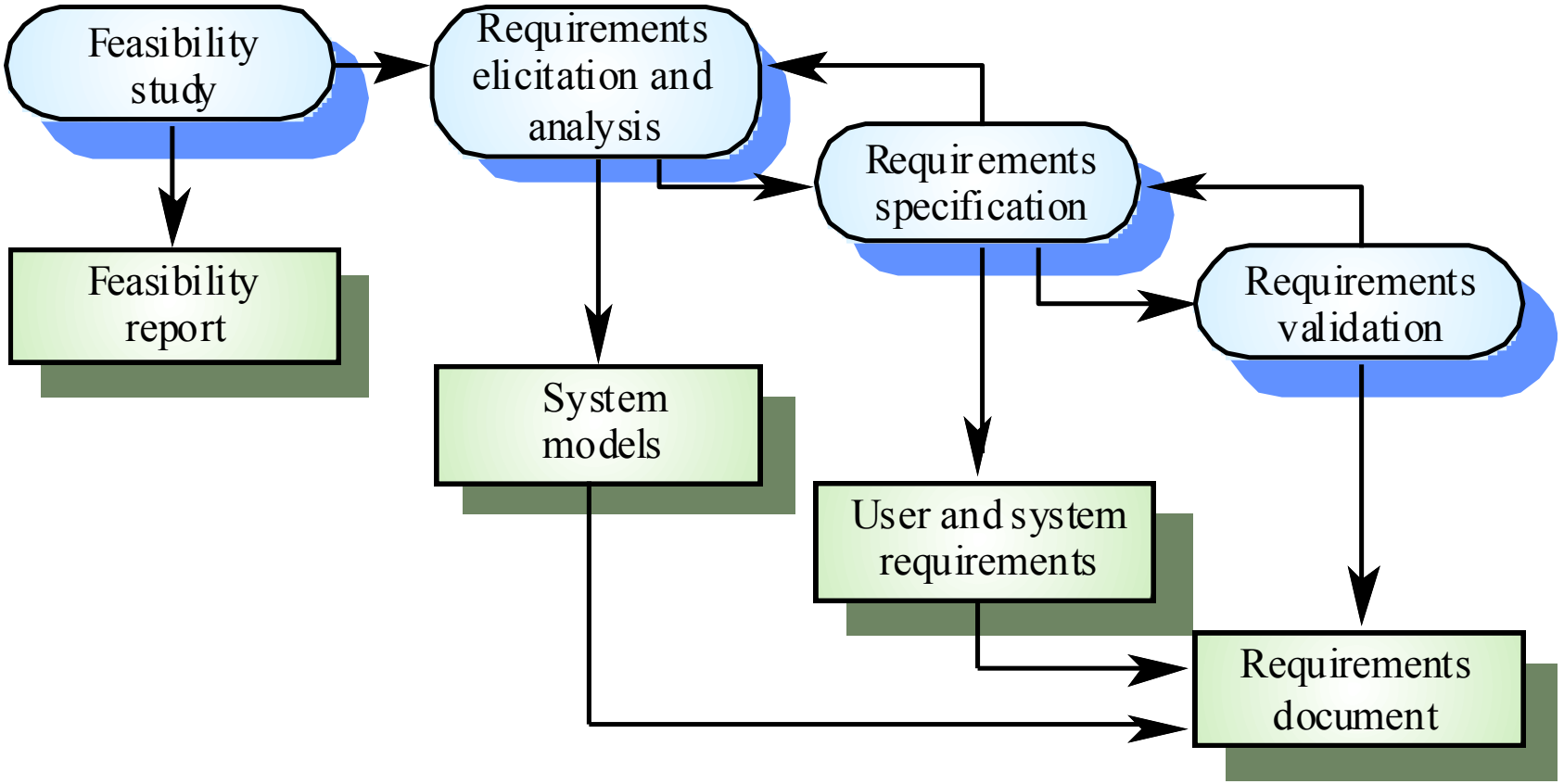
IMT2243 Systemutvikling

Forelesning 9 : forts. Kravspesifisering

- Forts. Objektorientert Analyse
  - System Sekvensdiagram, spesifiseringsteknikk innen OOA som viser interaksjonen knyttet til Use Case
- Struktureret Analyse
  - tradisjonell metode for kravspesifisering som er dels en forløper for og dels et alternativ til OOA
- Spesifisering av Operasjonelle krav
  - Kalles også «ikke-funksjonelle krav» eller «kvalitetsmessige krav»

Pensum: Komp.8 (SSD) og 9 + Sommerville kap4.1. (Operasjonelle) + foilsett (for SA).
(Sommerville kap. 4 og 5 gjennomgås senere)

Kravspesifiseringsprosessen



Figur 2.4 fra Software Engineering, Ian Sommerville

ObjektOrientertAnalyse (OOA)

En analysemetode :

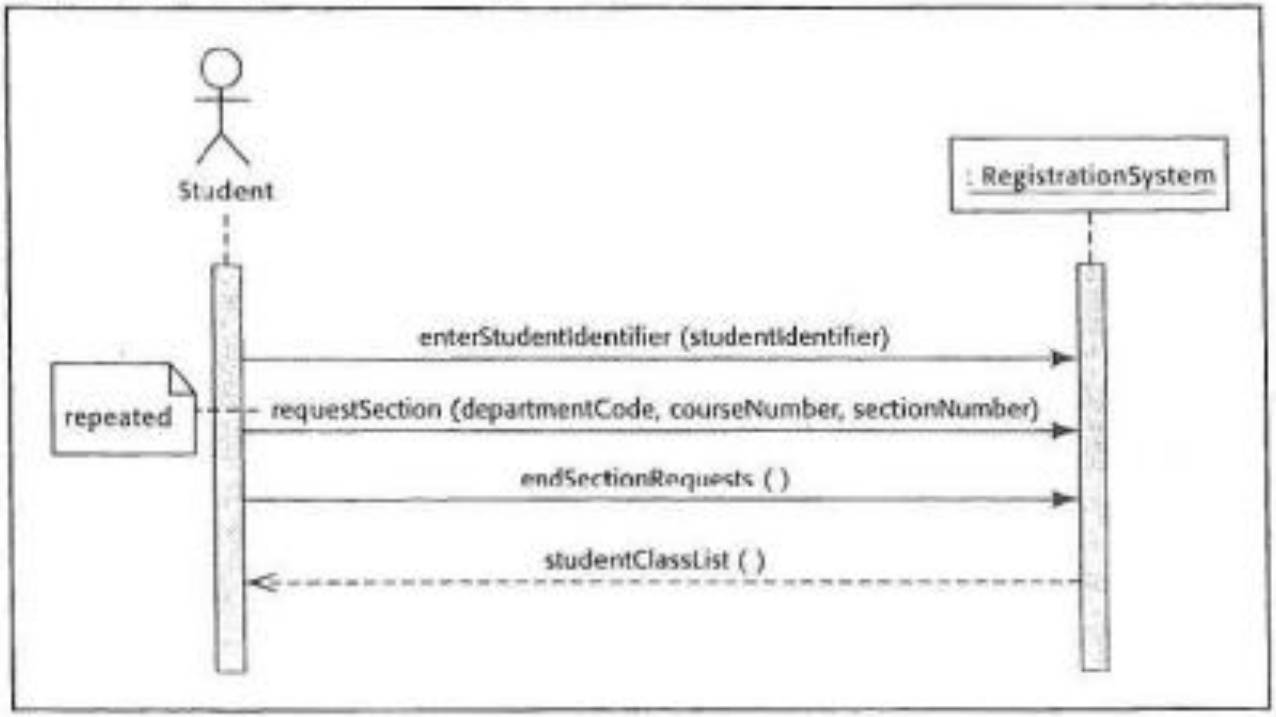
- Gir en beskrivelse av hvordan man legger opp arbeidet i analysefasen
- Betrakter Problemområdet fra et perspektiv der man ser både verdenen og programvaren (som skal ”speile” verdenen) som en samling samhandlende objekter
- Resulterer i Artefakter (informasjonsbærende modeller og beskrivelser) som bakes inn i selve kravspesifikasjonsdokumentet

- Sentrale artefakter vi ser på innen OOA er :
    - Use Case (diagram + tekstlig beskrivelse)
    - System Sekvensdiagram
    - Domenemodell / Konseptuelt klassediagram (senere forelesn)

SystemSekvensDiagram

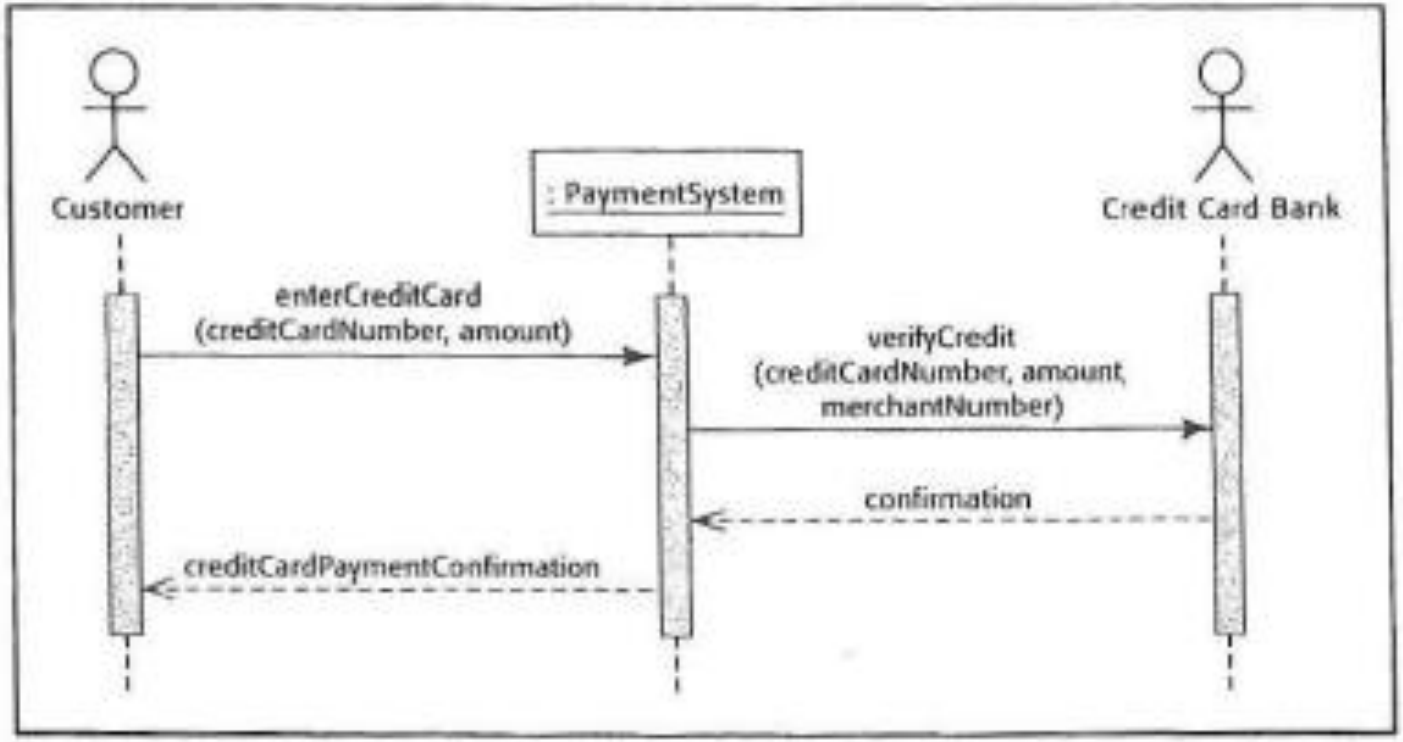
Pensum : Komp. Nr 8 : Vi gjennomgår s117 – 125 i kopien

FIGURE 4.23 System sequence diagram for the Register for Classes use case



SystemSekvensDiagram

FIGURE 4.25 System sequence diagram for the operation make Credit Card Payment



Senere skal vi se på Sekvensdiagram innen Design, dette er IKKE det samme som SystemSekvensdiagram!

Strukturert analyse

SA ble utformet på en tid dd fossefall-modellen hadde utstrakt anvendelse, og man oftest hadde rasjonalisering som hovedmål ved utvikling av nye IT-systemer. IT-systemene skulle i stor grad hjelpe til med å automatisere prosessering av informasjon.

Input – Process – Output (IPO) står i fokus for all kartlegging.

Metoden er fortsatt relevant å anvende i spesifiseringsarbeid spesielt når man arbeider etter en sekvensiell utviklingsmodell for transaksjons-orienterte applikasjoner.

Moderne programvarearkitekturer basert på tjenesteorientering bidrar nå til en renesanse for denne type modelleringsteknikker ved spesifisering av de funksjonelle krav.

Mer info på : http://en.wikipedia.org/wiki/Structured_analysis

forts. Strukturert analyse

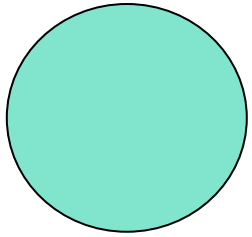
Innen Objektorientert Analyse (OOA) benyttes teknikkene

- Use Case (diagram + beskrivelser)
- System Sekvensdiagram
- Konseptuelle Klassediagram (Domenemodellering).

Innen Strukturert Analyse brukes følgende teknikker (systemmodeller/beskrivelser) :

- Dataflytdiagrammer (den sentrale teknikken i SA)
- DataDictionary
- ER-modeller (kom inn som supplement)
- Strukturert språk (nær pesudokode på prosessene)

DFD (DataFlytDiagram)



Prosess (funksjon)

- Bearbeider/manipulerer input om til output
- En prosessboble for hver funksjon i systemet
- Navngis ut fra hva den ”gjør”

DFD

Dataflyt (informasjonsflyt)

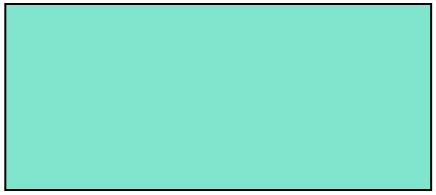
- Viser hvordan data/informasjon flyter i systemet.
- Vi fokuserer på logiske modeller og flyt av modeller. DFD anvendes også til å vise flyt av ”fysiske elementer”.
- Modellen viser ikke rekkefølgen på flytene, bare retning
- Navngis ut fra manipuleringen den gjør
- Detaljspesifiseres i DataDictionary

DFD

Datalager / register

- Viser en samling av data som må ligge lagret i systemet
- Viser ikke hvordan dataene skal lagres
- Dataene ligger passive inntil de blir kalt opp
- Unngå registre uten ”inn” og ”ut” flyt
- Lengst mulig ned i DFD-strukturen

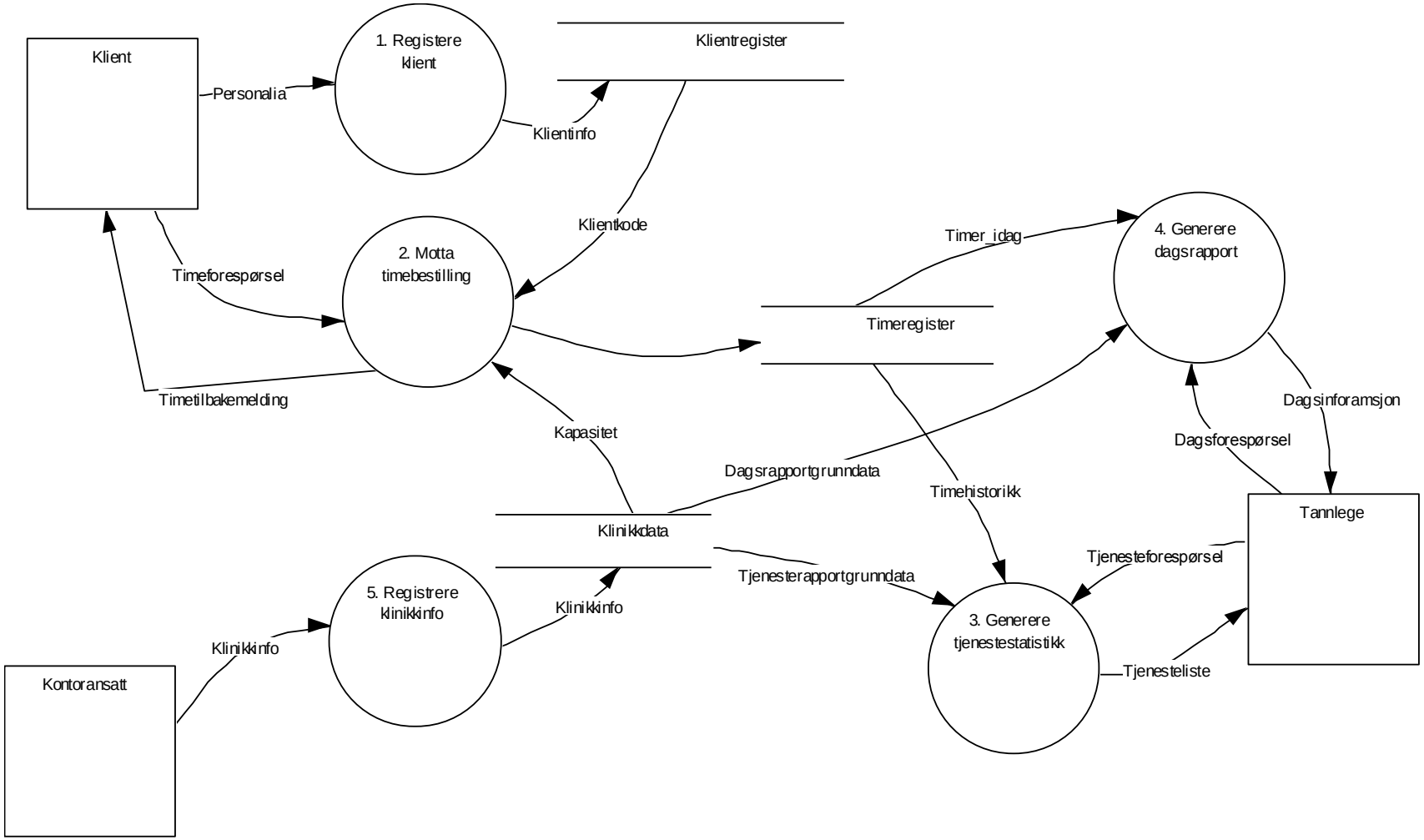
DFD



Terminator / Ekstern enhet

- Viser kilder til / mottaker av informasjonsflyt i vårt system
- Kan være roller, interessenter, andre systemer etc.

Eksempel på en SYSTEMMODELL :



Tips ved utarbeidelse av DFD

1. Hold modellen på en side - lesbart og forståelig
2. Ikke lag modeller med mange elementer
   - mindre enn 10 prosesser.

3. Bruk Dataflytdiagrammenes nivådelingsmekanisme
   Kontekstdiagram (systemboble + terminatorer),
   DFD1 (alle sentrale prosesser, infoflyter og registre)
   DFD2 (detaljering av mer «intrikate» enkeltprosesser)

4. Ikke forsøk å ”si alt” med modellen, vis det viktige

Kravspesifikasjonen

Kravspesifikasjonsdokumentet skal dekke :

- Funksjonelle krav
- Ikke-funksjonelle krav (operasjonelle, kvalitetsmessige)
- Grensesnitt mot brukere, andre systemer og enheter
- Avgrensninger

Dokumentet er sentralt som :

- Beslutningsgrunnlag for evt. videreføring av prosjektet.
- Vedlegg til en kontrakt mellom kunde og leverandør
- Grunnlag for detaljert tids-, kostnads- og ressursplan
- Utgangspunkt for alt designarbeid.
- Grunnlag for all testing

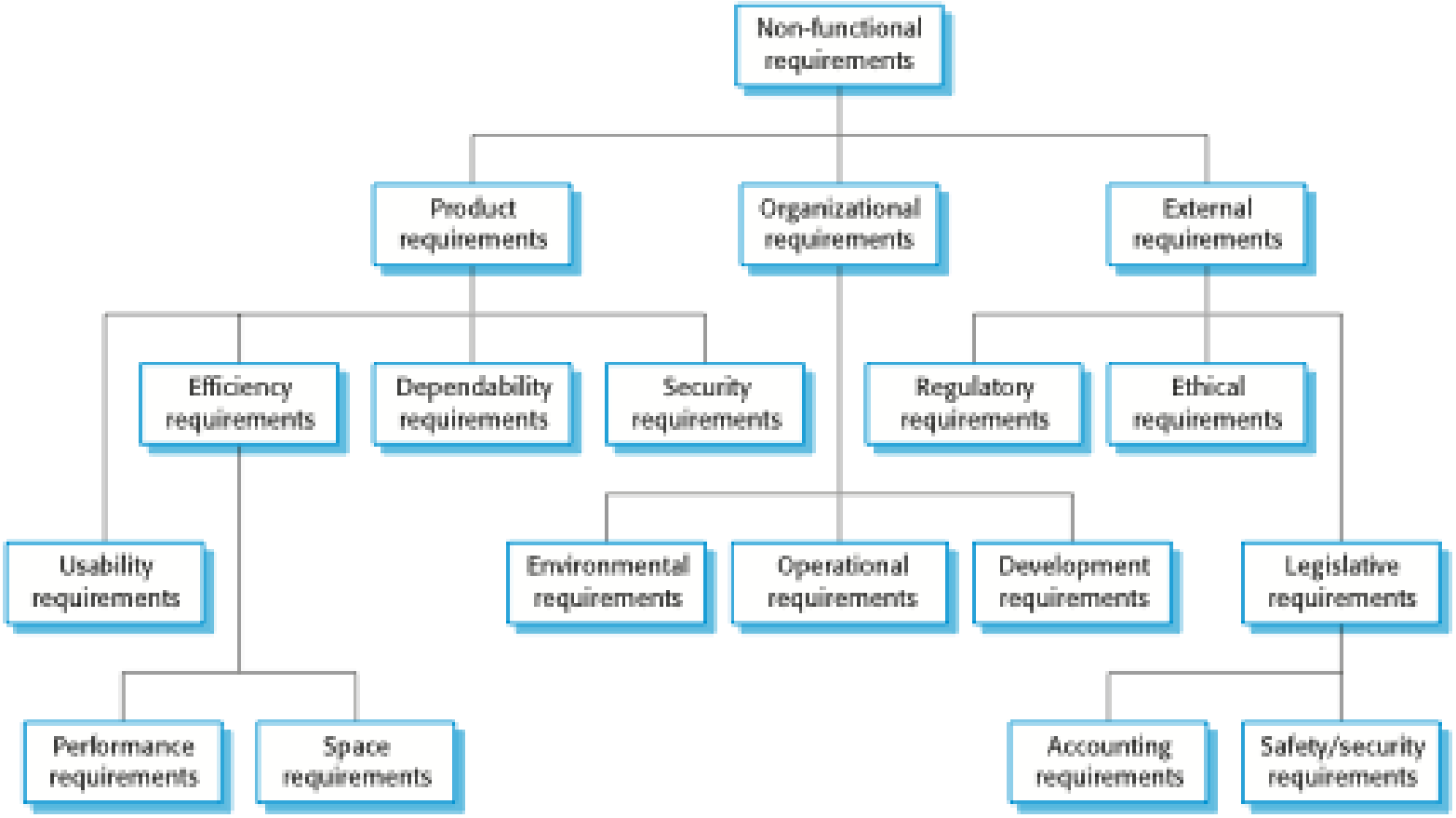
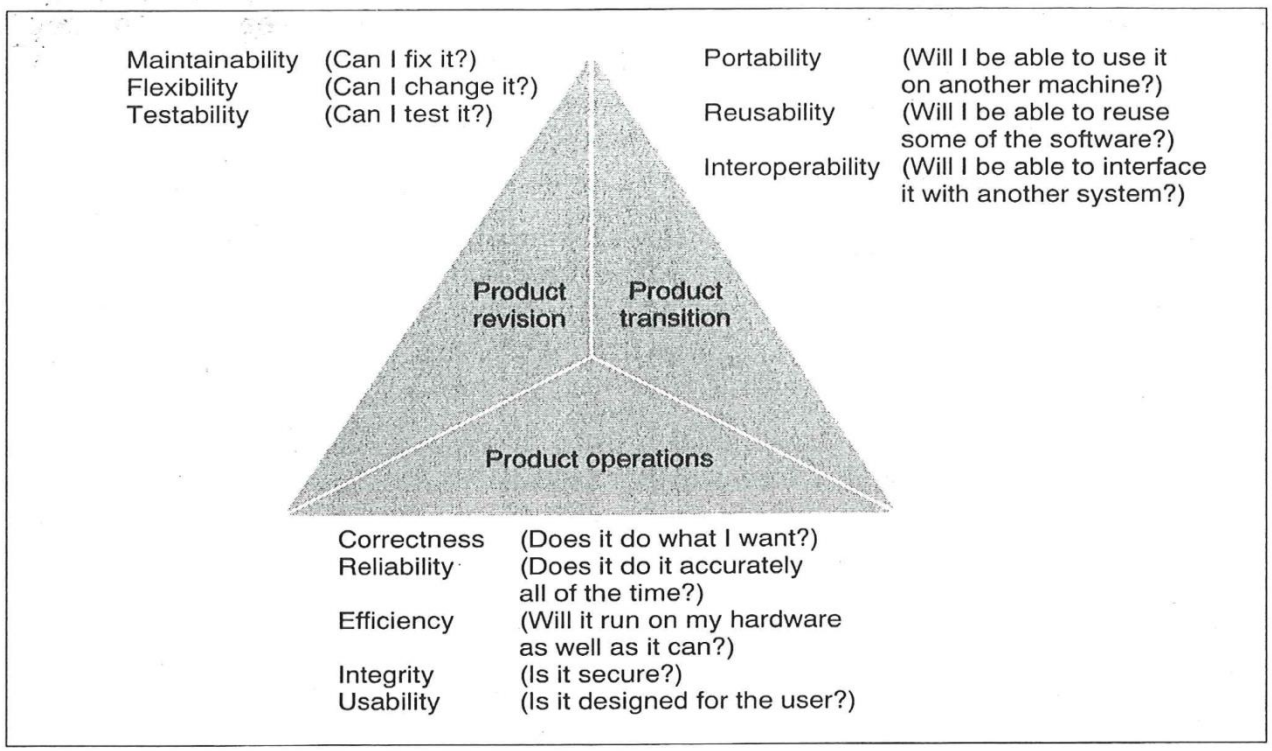
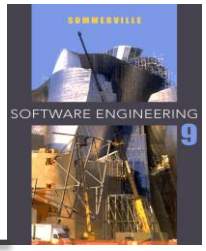


Figure 4.3 Software Engineering 9 ed. , Ian Sommerville

Roger Pressman : Software Engineering



Sommerville : Ways of writing a system requirements specification



Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract

Hva mangler vi nå relatert til kravspesifisering ?

Vi har nå lært teknikker for å fremskaffe funksjonelle og ikke funksjonelle krav :

- Use Case – dekker de funksjonelle kravene
- System Sekvens Diagram – avklarer interaksjon i «komplekse» tilfeller
- Operasjonelle krav – dokumenterer «run-time» kravene knyttet til den nye programvaren

Vi trenger :

- En modell som forteller om de sentrale «tingene» ute i den virkeligheten programvaren skal støtte og viser sammenhengene mellom elementene (Domenemodell/Konseptuelt klassediagram)
- En mal for hvordan vi skal sette opp et godt kravspek-dokument