Velkommen til

INF1050 – Systemutvikling

2009

Jo Hannay

Simula Research Laboratory & Institutt for Informatikk

INF1050 – Dagsorden 14. jan. 2009

- **Om kurset:**
  - **Læringsmål**
  - **Struktur, forelesere, undervisningsplan**
  - **Undervisningsmateriell**
  - **Gjennomføring**
  - **Obligatoriske oppgaver**
  - **Vurderingsform**
- **Introduksjon til systemutvikling**

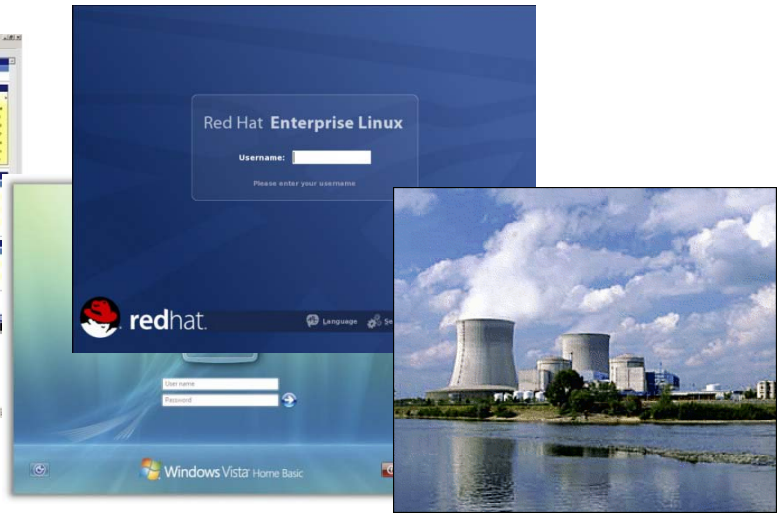
Læringsmål

❑ Du skal forstå hva det innebærer å utvikle et ”system”

Komplekst!

Hva da for ”system”?

- Software-system = Datasystem = IT-system



- Software = Programvare

Program = Programvare ?

- Systemutvikling = Softwareutvikling = Programvareutvikling

Læringsmål

- **Du skal forstå hva det innebærer å utvikle et ”system”**
  - **hvordan man fastlegger systemets egenskaper**
  - **hvilke rammer som gjelder for utviklingen**
  - **hvordan man lager selve systemet**
  - **hvordan man mest effektivt får tatt systemet i bruk**
  - **hvordan utviklingsprosessen styres**

Programmering?

Læringsmål

- **Du skal forstå hva det innebærer å utvikle et "system"**
  - **hvordan man fastlegger systemets egen**
  - **hvilke ramm**
  - **hvordan ma**
  - **hvordan man**
  - **hvordan utvik**

Systemutvikling fordrer tekniske, organisatoriske, psykologiske, sosiologiske, økonomiske og andre ferdigheter.

Systemutvikling er tverrfaglig!

- **Industrien ettersøker bedre kompetanse i alle disse feltene! Mange dårlige kandidater på jobbintervjuer!**

Kursets struktur og forelesere

Systemutvikling som helhet

1. Systemutvikling: motivasjon ..... Jo Hannay, Simula & Ifi
2. Systemutviklingsprosessen ..... Rune Steinberg, Visma Software AS
3. Prosjektledelse og prosjektarbeid ... Rune Steinberg, Visma Software AS

Kunde/leverandør/bruker-forhold

4. Kravhåndtering ..... Erik Arisholm, Simula & Ifi
5. Avtaler & kontrakter ... Jørgen Petersen, Promis AS
6. Estimering ..... Stein Grimstad, Simula
7. Jus & etikk .... Dag W. Schartum, Senter for Rettsinformatikk

15. Oppsummering & eksamenstips ...Erik Arisholm
16. Faglig sosial ettermiddag ..... Foreleserne og dere!

Detaljert undervisingsplan:
uio.no/studier/emner/matnat/ifi/INF1050/v09/undervisningsplan.xml

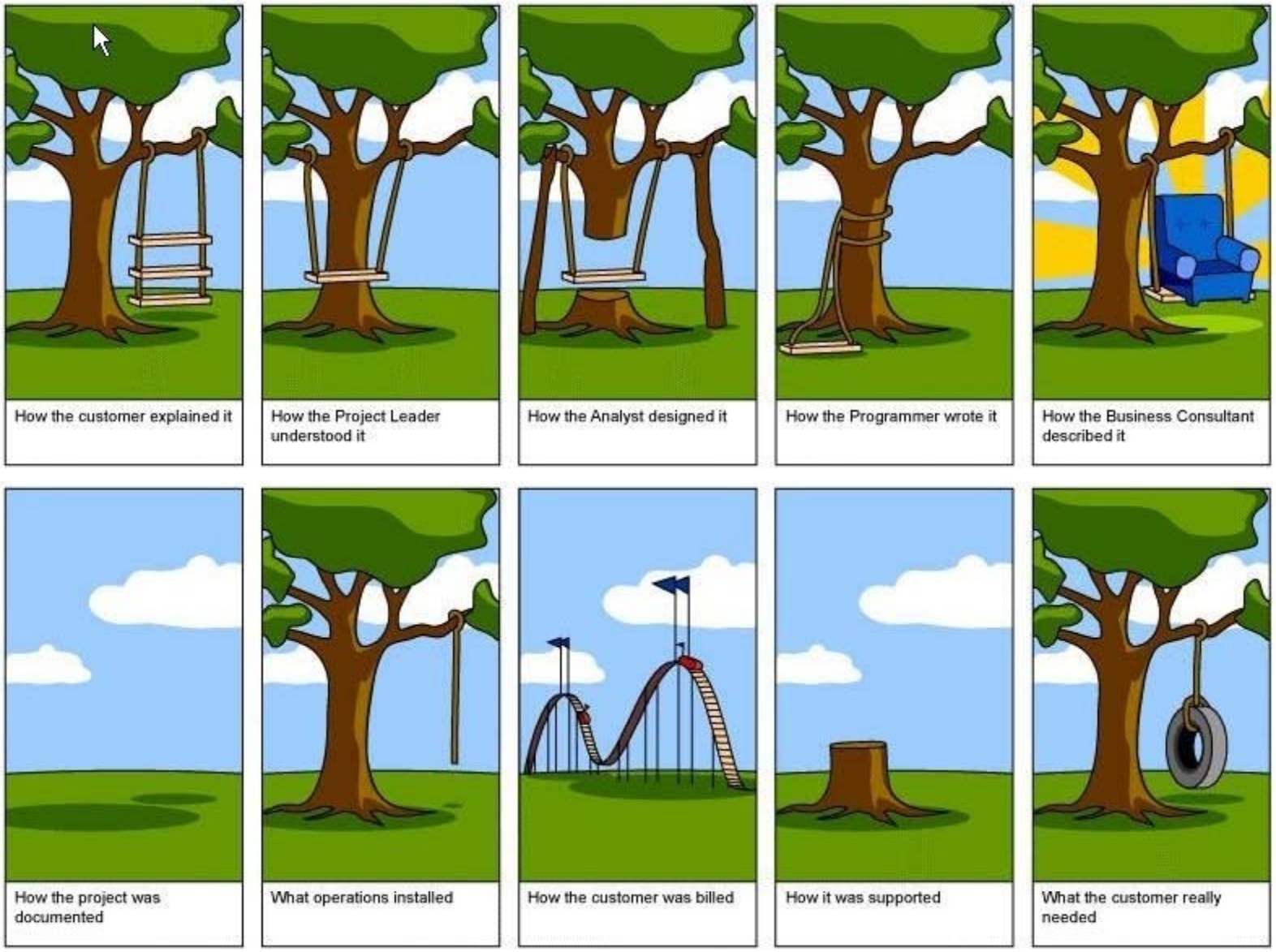
Systemets struktur og design

8. Modellering av krav med use cases ...Erik Arisholm, Simula & Ifi
9. Objektorientert analyse (2 forel.) ..... Erik Arisholm, Simula & Ifi
10. Persistens og databaser .....Erik Arisholm, Simula & Ifi
11. Arkitektur ..... Dag Lorås, Visma Software AS

Koding, validering og vedlikehold

12. Modellbasert utvikling med Genova ... Esito AS
13. Validering og verifisering (2 forel.) ..... Lionel Briand, Simula & Ifi
14. Konfigurasjonsstyring..... Hans Christian Benestad, Simula

Kunde **Leverandør** **Bruker**


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

x

Undervisningsmateriell

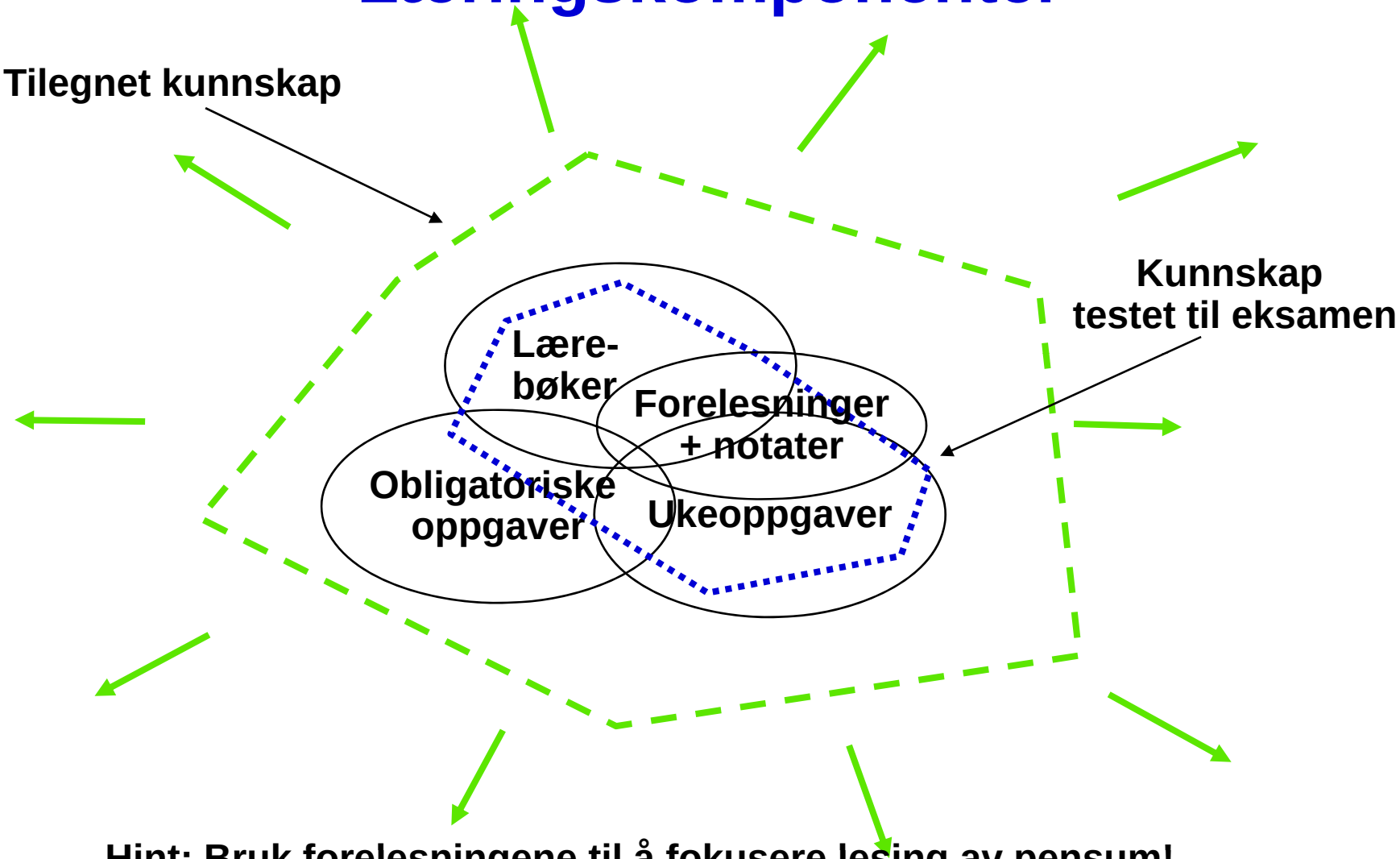
- **Lærebøker:**
  - ~~Gunnar Gurholt og Thor E. Hasle: Grunnleggende Systemutvikling (GS), 2003. Cappelen. ISBN: 82-02-19868-2. Pensum: Kapitler 1-11 og 14-19; samt emner 1-4 og 7-9~~
  - Thor E. Hasle: Systemutvikling – Applikasjoner og databaser. ISBN 9788202286057 Pensum blir fastlagt etter hvert….
- **Støttelitteratur:**
  - Leszek A. Maciaszek: Requirements Analysis and System Design (RASD), 3rd edition, 2007. Addison Wesley. ISBN: 978-0-321-44036-5.
    - Kapitler 1-4 og 7-9.
- **Forelesningsnotater, foiler og annet materiell:**
  - Legges ut senest mandagen før forelesning (som regel).
- **Ukeoppgaver til hver forelesning:**
  - Gjennomgås på gruppene i uka etter forelesningen. Legges ut senest etter forelesningen.
- **Tre obligatoriske oppgaver**
  - Orakeltjeneste i forbindelse med obligatoriske oppgaver!

Gjennomføring

- Gå på forelesninger og grupper!
- Studer undervisningsplan, les kapitlene i læreboka, samt forelesningsnotater og foiler (oppdateres hver uke) på http://www.uio.no/studier/emner/matnat/ifi/INF1050/v09/undervisningsplan.xml
- Forelesninger gir dere utfyllende stoff, samtidig som de vil fokusere på det vi mener er mest sentralt.
- Gjennom forelesningene vil vi forsøke å sette pensum i bedre kontekst av reelle systemutviklingssituasjoner.
- Forelesninger vil gi mer kritisk holdning til pensum og "guruers" påstander.
- Forelesninger kan være disiplinerende.
- Forelesninger gir mulighet til diskusjoner og spørsmål.

… som gjør det mulig å score ekstrapoeng på eksamen ;-)

Læringskomponenter



Hint: Bruk forelesningene til å fokusere lesing av pensum!

Obligatoriske oppgaver

- **Du skal**
  - **få øving i å kunne omsette kursets teori til praksis**
  - **opparbeide kjennskap til et utvalg plattformer og utviklingsverktøy**
  - **kunne anvende et utvalg systemutviklingsteknikker**
- **Oppgavene følger hovedtemaene (grovt sett):**
  - **Oblig 1: Idefasen, prosess, krav, kontrakter og estimering**
    - **Utlevering: 19.feb, innlevering: 6.mars**
  - **Oblig 2: Fra krav til UML analysemodell (Rational Rose)**
    - **Utlevering: 12.mars, innlevering: 3.april**
  - **Oblig 3: Prototyping av brukergrensesnitt (Genova). Forslag til databaseskjema.**
    - **Utlevering: 16. april, innlevering: 4.mai**
- **Du leverer individuelle besvarelser**

Du kan påvirke undervisningen!

- **Kontinuerlig kursevaluering og -forbedring**
  - **Når som helst:**
    - **Send e-post til inf1050@ifi.uio.no dersom du har negative eller positive tilbakemeldinger på kursets innhold eller gjennomføring**
    - **Eventuelt gi tilbakemeldinger til gruppelærerne, som vi har ukentlige møter med**
  - **Etter oblig 1:**
    - **5-minutters anonymt spørreskjema (detaljer kommer senere)**

Du har også et ansvar for undervisningen!

- **Vis omtanke og folkeskikk! Våre forelesere og dine medstudenter forventer at:**
  - **du er på plass i det forelesningen begynner**
  - **du ikke småprater med andre under forelesningen**
  - **du ikke spiller spill, surfer, chatter, emailer under forelesningen**
  - **du ikke leser avisen under forelesningen**
  - **du gjør alt du kan for å engasjere deg og følge med i forelesningen**

    - selv om det iblant kan være tungt!

Vurderingsform

- **Tre godkjente obligatoriske oppgaver**
  - **Studenter som tidligere har fått godkjent Inf1050 prosjektoppgave trenger ikke levere nye obligatoriske oppgaver.**
- **En skriftlig 3 timers eksamen (2. juni).**
  - **Alle trykte og skrevne hjelpemidler er tillatt.**
- **Informasjon om utsatt prøve (kontinuasjon) finner du her:**
  http://www.matnat.uio.no/studier/eksamen/kontinuasjon.html

e-post-adresser

- Faglige spørsmål og kommentarer: inf1050@ifi.uio.no
  (går til kursansvarlige: Erik Arisholm & Jo Hannay)
- Gruppespørsmål: inf1050-x@ifi.uio.no
  der x er gruppenummeret (går til gruppelæreren)
- Studieadministrative spørsmål: studieinfo@ifi.uio.no
  (går til studieadministrasjonen)

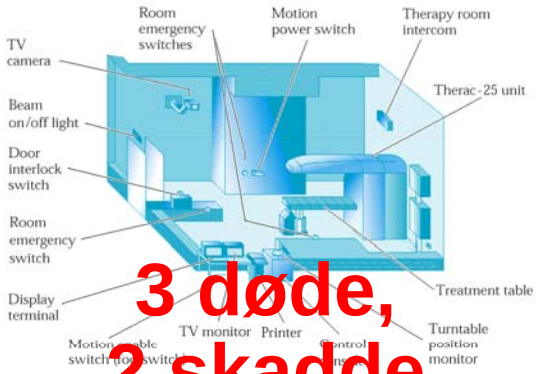
INF1050 – Dagsorden 14. jan. 2009

- **Om kurset:**

- **Introduksjon til systemutvikling**
  - **Motivasjon: Ting må gjøres bedre!**
  - **Ting *kan* gjøres bedre!**
  - **Metodikk**
    Software engineering (Industriell systemutvikling)

De-Motivasjon

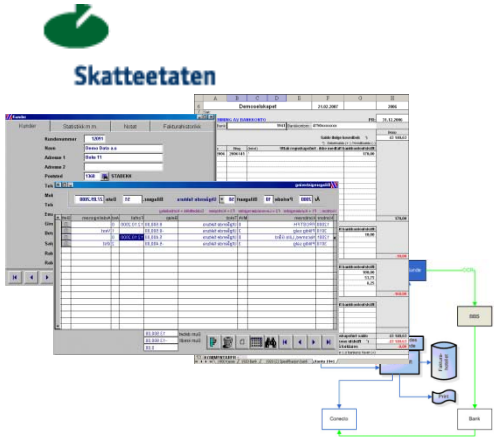
- Det utvikles og vedlikeholdes trolig programvare for mellom 10 og 40 milliarder kroner i året i Norge.
  - Ca. 75% av IT-prosjektene har overskridelser (2003)
  - I gjennomsnitt er kostnadsoverskridelsene på 30-40%
- En stor andel av prosjektene fullføres aldri – særlig de svært store prosjektene er utsatt for totalstopp eller langvarige forsinkelser.
- Ofte dårlig kvalitet.



Ting *kan* gjøres bedre!

- **F.eks.: *Utviklingsprosess* har stor påvirkning på kontrollen. Studier vi har gjennomført viste blant annet at (2003):**
  - **Fossefall: 55% overskridelse**
  - **Inkrementelle/iterative prosesser: 24% overskridelse**

- **Eksempel: SKARP-prosjektet**
  utvikle et skatteregnskapssystem



2003: 1 mrd., 7 år, ikke i drift

2008: i drift!

SKARP-prosjektet

- 1995: Dagens skatteregnskapssystem (Standardskatt) er over 20 år gammelt, Cobol-basert og vanskelig å vedlikeholde, og oppfyller ikke formelle krav til sikkerhet, kontroll og sporbarhet i slike systemer. Det koster også svært mye å drifte (50-60 mill kroner pr år).
- 1996: Prosjektet initiert. Dette er det største it-prosjektet direktoratet noensinne har igangsatt, med en kostnadsramme på nærmere 1 milliard kroner.
- 2000: WM-Data får fastpriskontrakt på levering av det nye skatteregnskapssystemet (SOFIE) for Skattedirektoratet.
- 2002: Testingen av leveransene fra WM-data skulle startet tidlig våren 2002, og skulle etter planen settes i drift høsten 2002. Det er forsinkelser i prosjektet. Rykter om at WM-data allerede utvikler ”gratis”.
- 2003: Skattedirektoratet hever avtalen med WM-Data. Skatteetaten mener at årsaken til forsinkelsene i SKARP-prosjektet først og fremst skyldes det uføre kontrakten med VM-data medførte.
  - VM-data taper prestisje, 250 millioner kroner, og 28 ansatte måtte gå.
- 2003: Ny avtale inngås med Cap Gemini, basert på
  - Todelt kontrakt: SOFIE Basis (kjernen) og SOFIE Innføring (brukergrensesnittet))
  - PS2000 kontraktstandarden (foreleses av Jørgen Petersen: Avtaler og kontrakter) og
  - iterativ/inkrementell prosess (foreleses av Rune Steinberg: Utviklingsprosesser).
- 2005: Pilotkommuner i drift (Stor bidragsyter for å bedre kvaliteten på systemet. Brukerstøtte sentralt. Stor utfordring som må løses: konvertering av data fra gammelt system).
- 2006: Cap Gemini inngår tre kontrakter om sluttleveranser (utvidet funksjonalitet og feilrettinger). Svært fleksibel kontraktsform i forhold til hvilke utvidelser og feilrettinger som skal med i hvilken release.
- 2007/2008: Alle skatteoppkreverne tar i bruk systemet i løpet av 2007, med unntak Oslo kemnerkontor som ikke vil ta systemet i bruk før 2008. SOFIE er basert på Oracle Applications og over 1000 egenutviklede programvaremoduler.

Evidens-basert Metodikk: Software Engineering

- **Software engineering (industriell systemutvikling) omhandler teorier, metoder og verktøy for spesifikasjon, design, konstruksjon og vedlikehold av programvare.**
- **Tar i betraktning de menneskelige aspektene i samspill med de teknologiske aspektene!**
- **Er ment å bidra til at vi lager bedre systemer, raskere, med færre ressurser og på en mer forutsigbar måte.**
- **Basert på ingeniørprinsipper (systematiske metoder) med fokus på:**
  - **Planlegging og forutsigbarhet** (i motsetning til "ta den tiden som trengs")
  - **Oppdeling og strukturering av problemer i mindre komplekse bestanddeler** (i motsetning til "prøv og feil")
  - **Modularitet og gjenbruk** (i motsetning til "lag alt fra bunnen av hver gang")
  - **Abstraksjon og modellering** (i motsetning til "koden er systemet")
  - **Systematisk kvalitetssikring** (i motsetning til "gjør som du vil bare produktet blir bra")

5 hovedprinsipper

Planlegging og forutsigbarhet

- **Veldefinerte, repeterbare og planlagte aktiviteter**
  - **Alle personer vet *hva* de skal gjøre, *hvordan* det gjøres (standarder/metoder/verktøy), hva de skal levere og når det skal leveres.**
- **Prosjektplaner og -rapportering**
  - **Ressursplaner: Kostnadsrammer, personal, utstyr**
  - **Tidsplaner: Estimering, milepæler, aktivitetsnettverk**
- **Kvalitetsplaner og -rapportering**
  - **Sjekklister, inspeksjoner, testplaner, testresultater …**
  - **Rutiner for å håndtere endringsforespørsler, sporbarhet, …**
- **Men graden av planlegging og formalitet i systemutvikling er et kontroversielt tema**
  - **Lettvektsprosesser (f.eks. eXtreme Programming – XP) vs sekvensielle prosesser (f.eks. Fossefall) eller mer formelle metoder (f.eks. Model Driven Architecture - MDA).**

Oppdeling og strukturering av problemer i mindre komplekse bestanddeler

- **Oppdeling i for eksempel**
  - **Tid (faser)**
    - **PS2000 kontraktstandarden: *Behovsfase, Løsningsbeskrivelse, Iterativ konstruksjonsfase, Godkjenningsfase***
    - **Timeboxing/tidsavgrensning**
  - **Oppgaver (aktiviteter og tilhørende leveranser):**
    - ***Analyse, design, programmering, testing*, …**
  - **Tekniske aspekter**
    - ***Kvalitetsaspekter, funksjonalitet, moduler, komponenter***
  - **Modeller på forskjellige abstraksjonsnivåer**
    - ***Kravspesifikasjoner* versus *Objektorientert analyse* versus *Detaljert design* versus *Kode***

Modularitet og gjenbruk

- **Datasystemer deles opp i mindre delsystemer (komponenter, moduler, aspekter) slik at:**
  - **Hvert delsystem implementerer et veldefinert problem (høy kohesjon) og**
  - **Man forsøker å redusere avhengigheter på tvers av delsystemer (lav kobling)**
  - **For "nyvinninger" innen dette, se aspekt-orientert utvikling**
    - **Eksempler på aspekter:** sikkerhet, kontroll og sporbarhet (se SKARP-prosjektet)
- **Modularisering**
  - **Muliggjør gjenbruk innen et prosjekt eller på tvers av prosjekter**
  - **Letter arbeidsfordeling og samarbeid**
  - **Muliggjør inkrementell utvikling**

Abstraksjon og modellering

- Identifiser de viktigste momentene og ignorer detaljer som er irrelevante for å løse et gitt problem
- Modeller er en type abstraksjon: spesifikasjoner og designmodeller skjuler irrelevante programmeringsdetaljer
  - Selve programmet kan også ses på som en presis modell av hvilke oppgaver som skal gjøres og hvordan, som deretter oversettes til maskinkode slik at datamaskinen kan utføre dem
- I Inf1050 vil dere lære hvordan Unified Modeling Language (UML) kan brukes til å
  - spesifisere kravene til et system og gjøre en analyse av hvordan disse kravene kan realiseres i et objektorientert programmeringsspråk
  - definere en database for lagring av dataene
  - generere prototyper av blant annet brukergrensesnittet (vha Genova) for å få tidlige tilbakemeldinger fra potensielle brukere

Systematisk kvalitetssikring

- **Validering og verifisering**
  - **Validering: Har vi spesifisert systemet riktig?**
  - **Verifisering: Lager vi det spesifiserte systemet riktig?** x
- **Endringshåndtering og konfigurasjonsstyring**
- **Kundeinvolvering**
- **Inkrementell og iterativ utvikling**
  - **Reduserer risiko ved at man kan levere og evaluere (validere og verifisere) enkelte delsystemer**

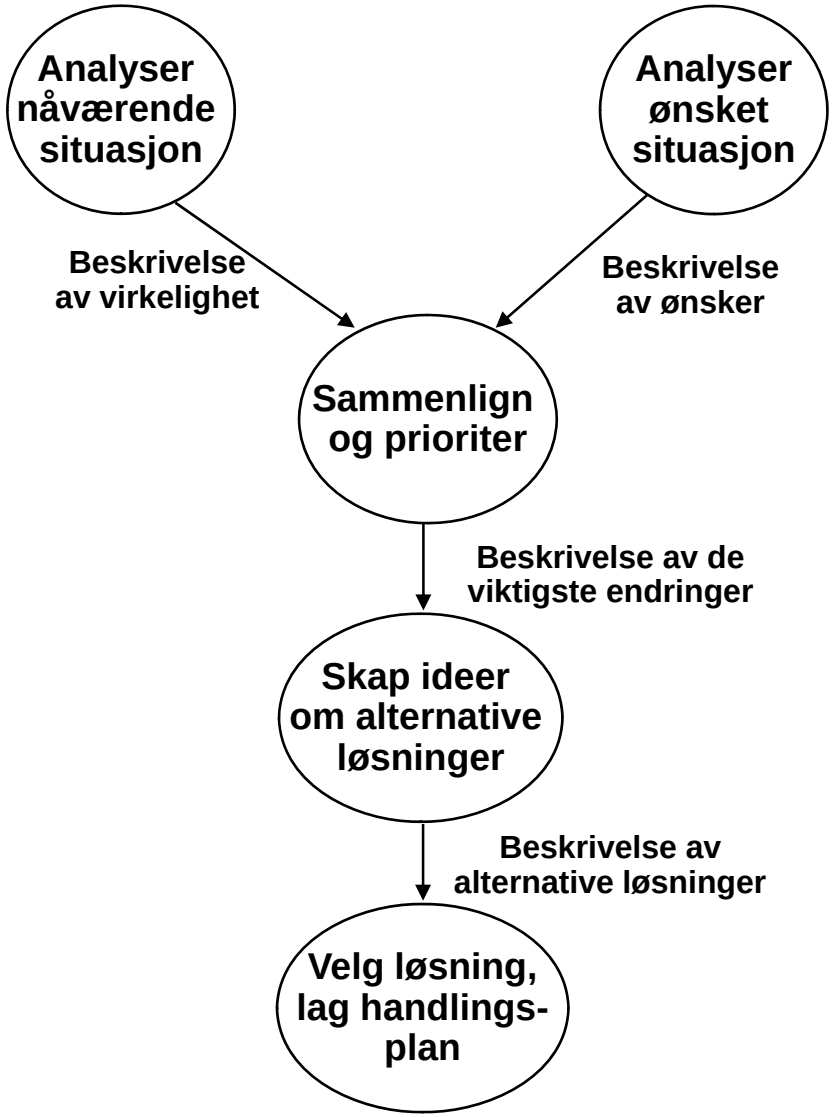
Systemutviklerens arbeid (GS kap. 1)

Noen ”myter” om systemutvikling

- **Myte 1: Systemet må tilpasse seg datamaskinens krav**
  (bruke få ressurser, være tilpasset fysisk arkitektur)
  - Informasjonssystemet skal være tilpasset bruker og deres krav!
  - Nøkkelord: Evolusjonær utvikling, evaluering av prototyper, brukergrensesnittdesign
- **Myte 2: De beste datasystemer er de du selv bygger opp fra grunnen (”Not Invented Here”-syndromet)**
  - Studier viser at det er overraskende lite gjenbruk av funksjonalitet eller kode mellom open-source prosjekter!
  - Undersøk først om det finnes standardprogrammer (eller et sett med standardkomponenter) som allerede dekker (deler av) behovene, eller som enkelt kan tilpasses til å dekke behovene
  - Nøkkelord: komponentbasert utvikling

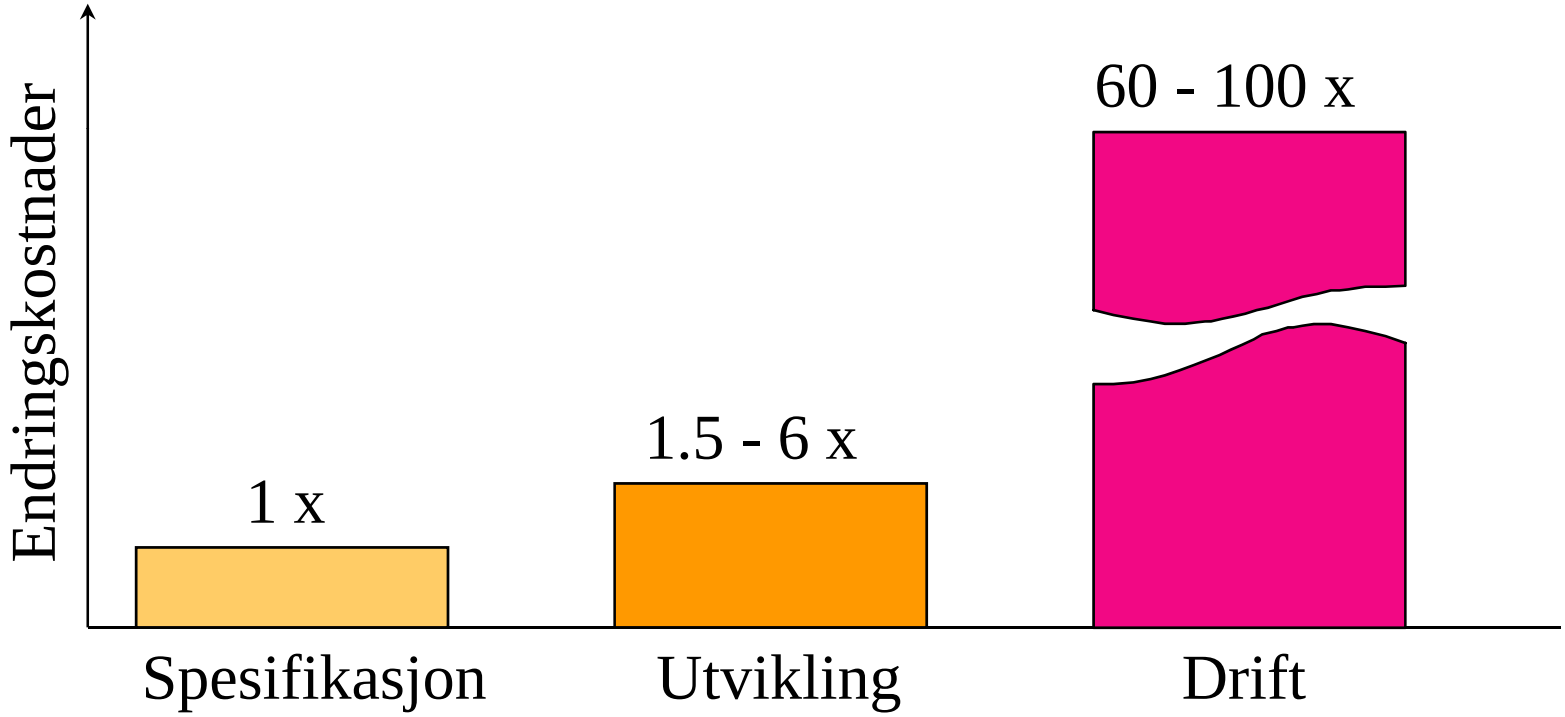
Myte 3: Datasystemet skal automatisere gamle rutiner

- Systemutvikling dreier seg også om å identifisere forbedrings-muligheter og introdusere nye løsninger (rutiner og prosesser) for å oppnå bestemte mål. Ofte vil innføringen av et nytt datasystem omlegge rutinene i en bedrift totalt!
- Nøkkelord:
  - Målanalyse
  - Business process reengineering



En myte til…

- **Myte 4: Programvare er så fleksibel at den kan alltids endres senere**



Noen flere myter om systemutvikling

- **Myte 5: Løsningen ligger i å bruke riktige verktøy**
  - **"A fool with a tool is still a fool"**
- **Myte 6: Ansett flere programmerere hvis prosjektet er forsinket**
  - **Brook's Mythical Man-Month: "adding people to a late software project makes it later":**
    - **Opplæring**
    - **Antall kommunikasjonskanaler = n(n-1)/2**


WE NEED TO FINISH YOUR PROGRAM TWICE AS FAST, SO I'M ADDING A PERSON TO HELP YOU.
YOU MIGHT NEED TO TRAIN HIM A LITTLE BEFORE HE'S PRODUCTIVE.
WARNING! WARNING! DR. SMITH
S. Adams
3-7 © 1995 United Feature Syndicate, Inc. (NYC)
TELL ME AGAIN WHAT THE BIG GLOWING THING IS.

God reise!