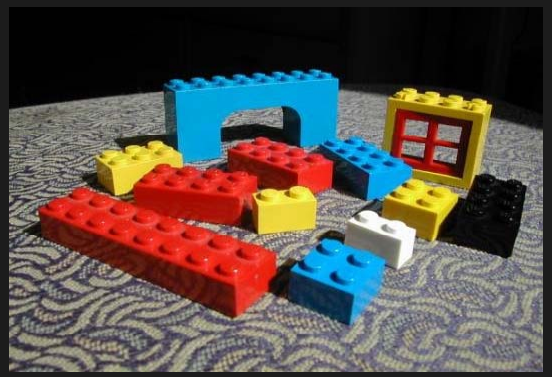
Endringshåndtering og konfigurasjonsstyring av programvare (SCCM)

Hans Christian Benestad

Simula Research Laboratory



Denne forelesningen beskriver SCCM's rolle under programvare-evolusjon

Behovet for endringer
(og endringskontroll...)

Endringsprosessen
(med knytninger til
tidligere forelesninger)

Håndtering av endringer
i systemkomponenter

Målet for SCCM er å holde orden på endringer under utvikling og evolusjon

Hvorfor ble dette endret? Ser helt feil ut. Jeg endrer tilbake.

Utviklere

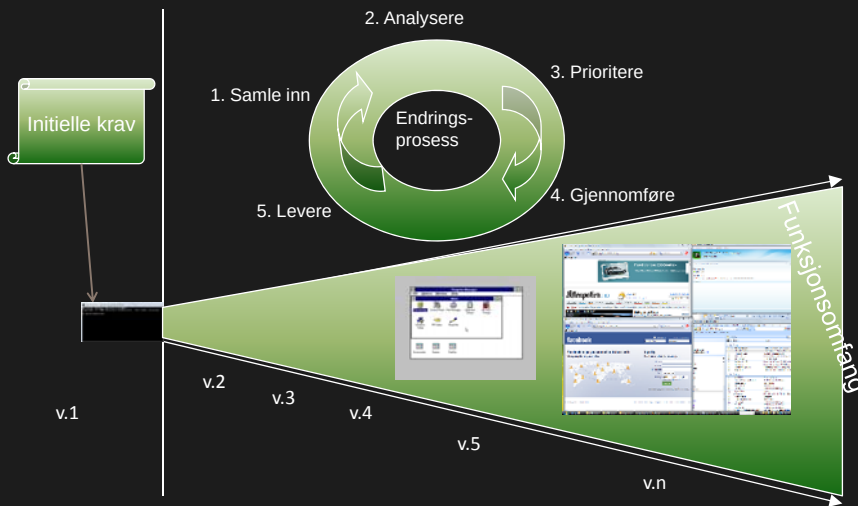
Er det noen som vet om vi har rettet denne feilen?

Prosjektet

Denne feilen var rettet men dukker nå opp igjen etter at jeg gikk over til Linux!

Brukere

Evolusjonsfasen kan være lang sammenlignet med initiell utvikling



Det er nødvendig og ønskelig å endre programvare

An *E*-type system must be continually adapted else it becomes progressively less satisfactory in use
- Manny Lehman 1974

Embrace change
- Kent Beck 1999

Noen årsaker til endringsbehov:

- Feilretting
- Tilpasning til nytt kjøremiljø
- Nye brukerkrav

"Dimensions of software maintenance " – Swanson 1976

Vellykket evolusjon forutsetter kontrollert håndtering av endringsønsker

Hver endring gjennomgår en systemutviklingsprosess i mikroformat

1. Samle inn

2. Analysere

3. Prioritere

Endrings-prosess

4. Gjennomføre

5. Levere

En *endringsprosess* beskriver kommunikasjonsveier, arbeidsflyt, ansvar og beslutningsprosesser

Endringskontroll av programvarekomponenter er sentralt i endringsprosessen

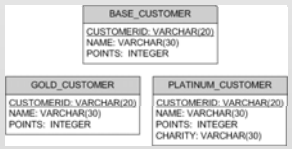
Krav

Modeller

Versjonskontrolldatabase

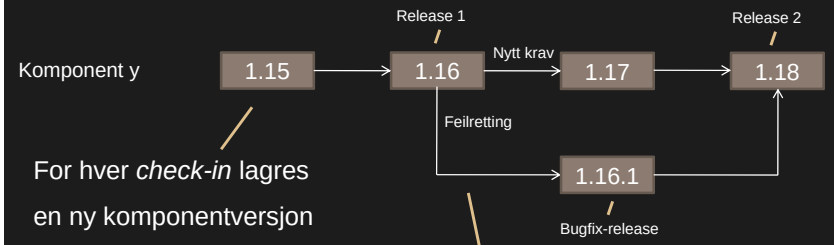
Verktøy og biblioteker

Kildekode



Databaseskjema

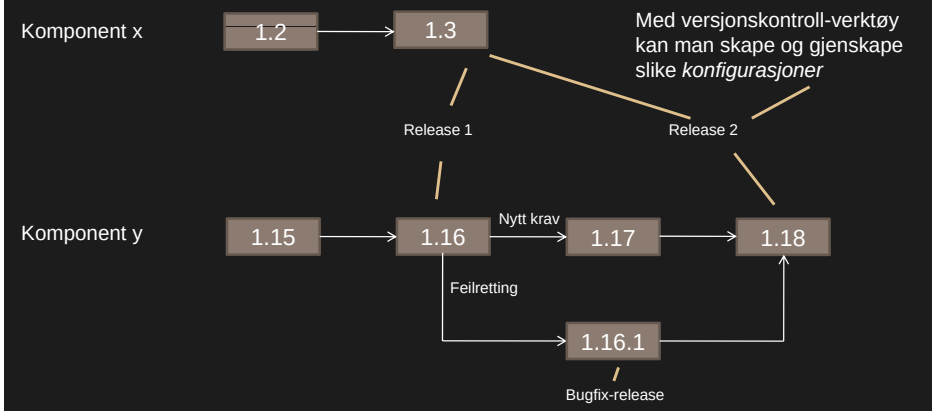
Et *versjonstre* inneholder hver komponents historikk



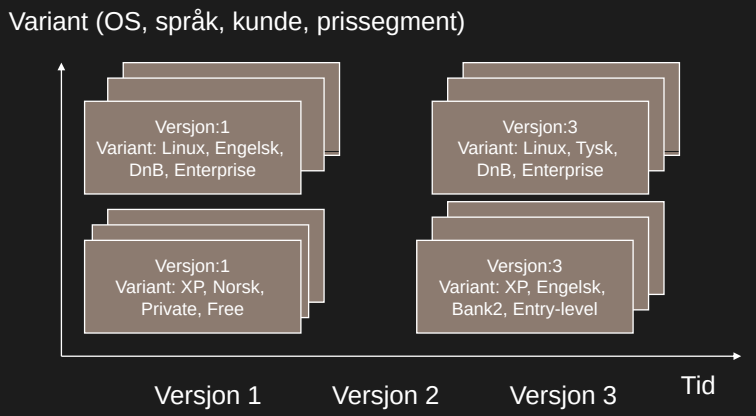
For hver *check-in* lagres en ny komponentversjon

Vanligvis bare en rett linje (trunk, mainline), men forgreninger kan være nødvendig

I en *konfigurasjon* inngår nøyaktig én versjon av hver komponent



Varianter og *versjoner* er ortogonale begreper



Varianter kan håndteres *dynamisk* eller *statisk*

Dynamisk: Programvaren tilpasser seg omgivelsene under kjøring

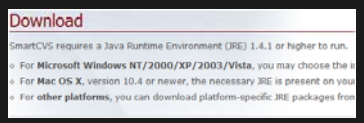
Eksempel: Program leser miljøvariabel LANG, og hent brukermeldinger fra fil med navn LANG.txt

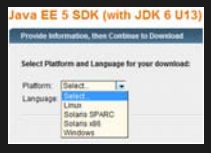
+En felles versjon til alle brukere

Statisk: Den kjørbare programvaren lages i forskjellige varianter

Eksempel: Inkluder ulike kildefiler under bygging av ulike varianter

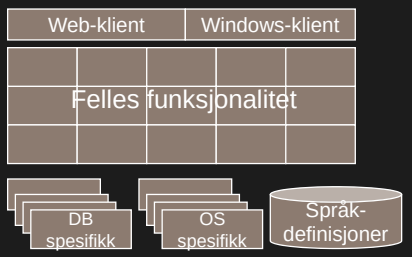
+Kan gi enklere/mer effektiv kode
- Kompliserende SCCM





En god arkitektur forenkler varianthåndtering

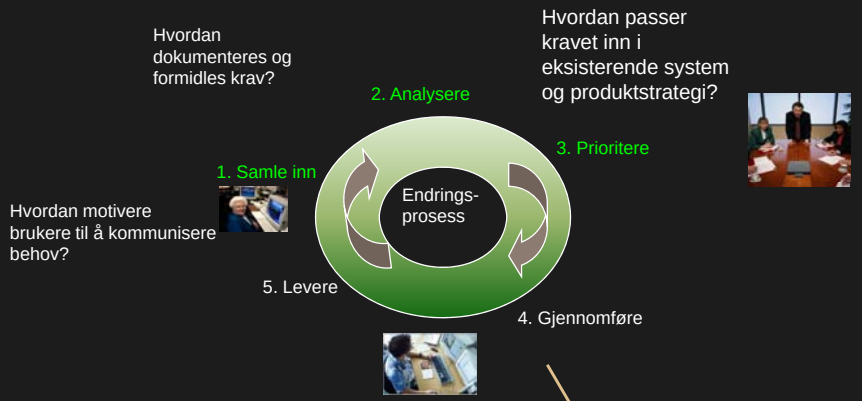
Tre-lags arkitektur er et godt utgangspunkt



Særegenheter for GUI, operativ- og databasesystem i adskilte komponenter

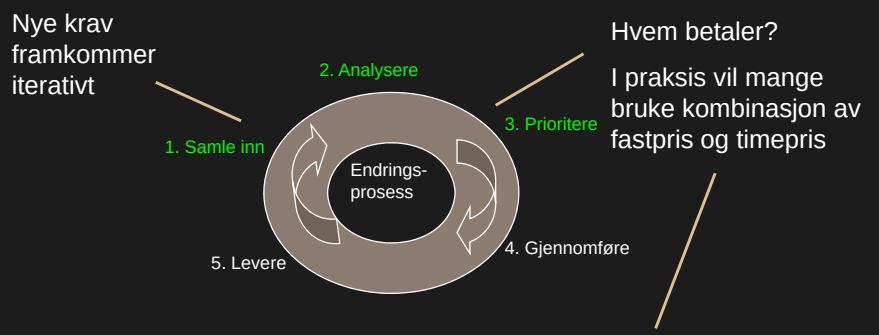
Data-definert oppførsel (eks. språk,kundesegment)

Kravhåndtering er viktig i endringsprosessen



Populære smidige metoder anbefaler en iterativ kravprosess også under initiell utvikling!

Kontraktstyper som forutsetter veldefinert omfang er vanskelig å bruke under videreutvikling



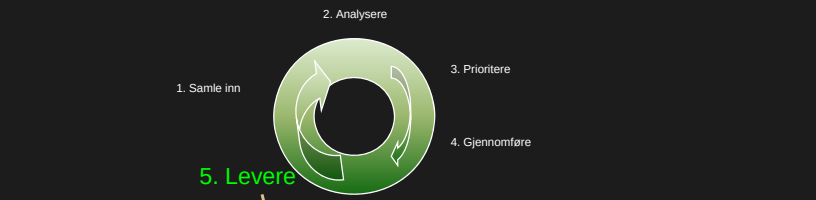
Kommersielle avklaringer må integreres i endringsprosessen

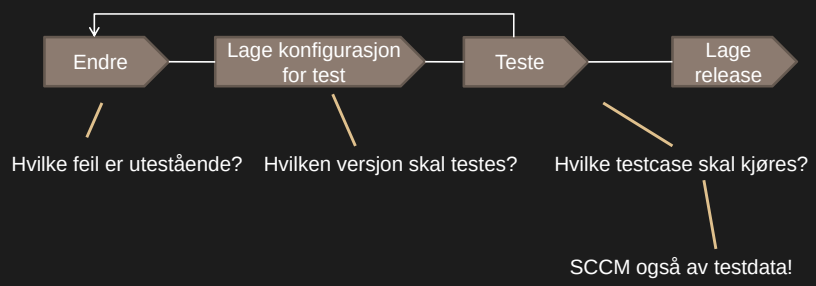
Fra PS2000:

E 2.2 Omfang av garantiytelsene

Garantien dekker feilsøking og utbedring av Feil i Leveransen. Dersom det er tvil om Leverandøren er ansvarlig for feilsituasjonen, skal Leverandøren likevel kontinuerlig bidra i feilsøkingen, frem til det er dokumentert at Kunden eller en tredjepart er ansvarlig for feilsituasjonen og således også for utbedring. Alle Feil som er gjenstående fra Godkjenningsprøven skal utbedres i denne perioden.

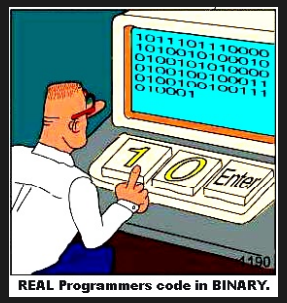
SCCM er sentralt ved testing før leveranse





God SCCM kommer brukerne til gode

God SCCM kommer utviklere til gode



Hvilke endringer har jeg ansvar for?

Hvilken versjon og variant gjelder endringen?

Hvorfor er denne programkoden endret?

God SCCM kommer prosjektet til gode

Når kan vi slippe neste versjon?

Hva må testes før neste versjon?

Hva må oppdateres i brukerdokumentasjonen?

Kognitiv kapasitet er en kritisk begrensning i systemutvikling

- God SCCM frigjør kapasitet til kreativt arbeid

Når er SCCM irrelevant?

Én utvikler lager én versjon av ett system

Men så populært dette ble da

MS-DOS R.I.P.

Opprinnelig kodebase lever ofte lenge!

MS-DOS passed away Thursday, October 25, 2001, at the Marriott Marquis Hotel on Times Square in New York City.

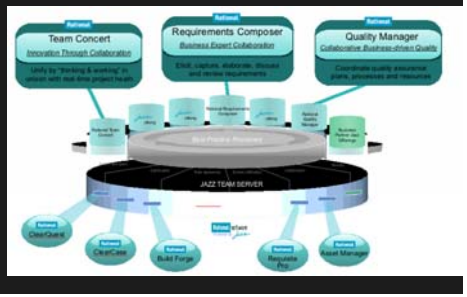
MS-DOS was born in August 1980, in Tukwila, Washington, the creation of Tim Paterson and the Seattle Computer Company. Initially called QDOS 0.10 (short for "Quick and Dirty Operating System"), MS-DOS

Dosering av SCCM må tilpasses behov

Uformelle vs. formaliserte rutiner.

Enkle vs. state-of-the-art verktøy

Subversion klient — Subversion server



Behovet er større i store prosjekter

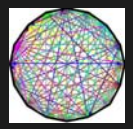
- Antall kommunikasjonsveier øker kvadratisk (n^2-n)/2

n=3 — 3

n=5 — 10

n=8 — 28

n=16 — 120



…og i desentraliserte prosjekter

Behovet også større for

”Innkapslet” programvare

Sikkerhetskritisk programvare

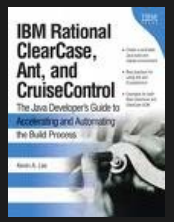
Virksomhetskritisk programvare

Bruk av verktøy for versjonskontroll

MKS

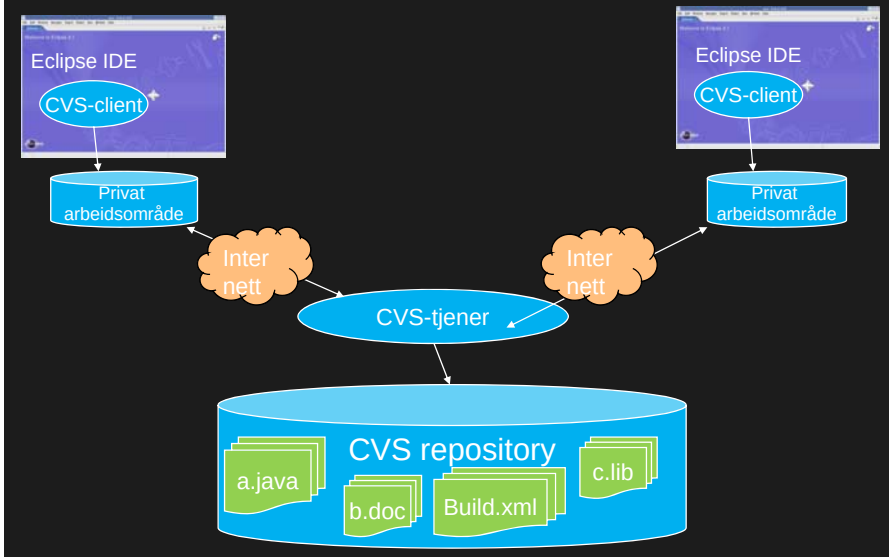
PERFORCE SOFTWARE

Telelogic Synergy Suite

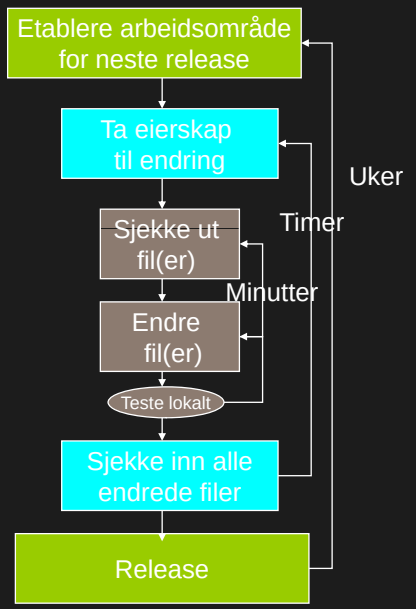


SUBVERSION

Typisk arkitektur i versjonskontroll-systemer



Endringssyklus sett fra utvikler

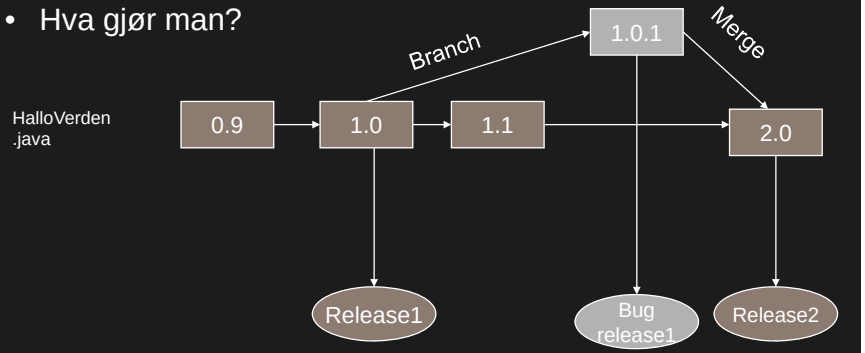


Arbeidsområde (workspace/view) er en utviklerspesifikk konfigurasjon

- For å kunne gjennomføre, bygge og teste må hver utvikler ha sitt personlige arbeidsområde
- Et typisk innhold i arbeidsområder er
  - Siste versjon i Mainline
  - Overstyrt av siste versjon i en aktuell forgrening (for eksempel BugRel1)
  - Overstyrt av egne lokal endringer som ikke er sjekket inn enda
- Holdes synkronisert med innholdet i repository
  - Eksplisitt via kommando ”Update”
  - Eventuelt automatisk (støttes av noen verktøy)

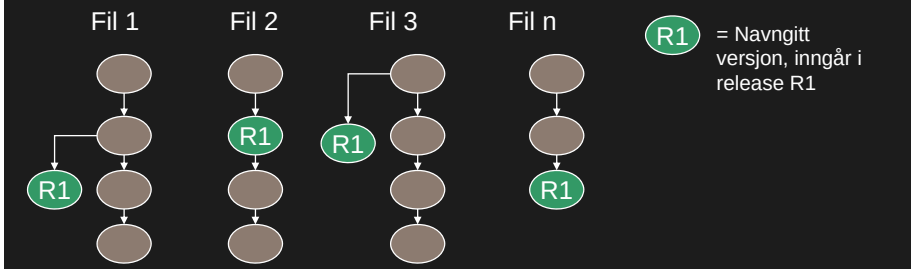
Løpende utvikling må kunne skilles fra feilretting

- Scenario:
- Release1 er sluppet, og utvikling pågår for release2
- Kunden opplever kritisk feil i Release1
- Hva gjør man?



Navngiving (label/tag) gjør det enkelt å gjenskape konfigurasjoner

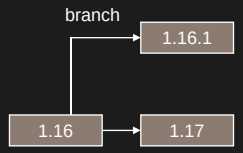
All kode som inngår i en release assosieres med et navn, for eksempel R1 (> cvs tag -R R1)



R1 kan da enkelt gjenskapes, med kommando som (> cvs checkout -r R1)

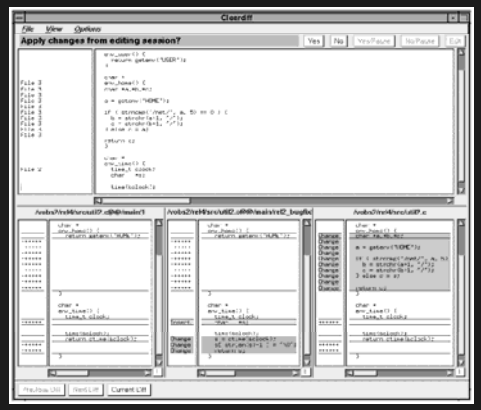
Med forgrening kan man jobbe med flere utgaver av samme komponent

- Aktuelt ved
  - Skille langsiktig utvikling fra kortsiktige releaser
  - Eksperimentell utvikling
  - Utvikling av større features
  - Samtidig utvikling i samme fil
- Kost/nytte må vurderes
  - Forgrening for hver logiske endring er som regel overkill
- Langlivede forgreninger bør unngås



Fletting (merge) slår sammen forgreninger

- 3-veis merge er vanlig
- Konflikter må håndteres manuelt, med støtteverktøy

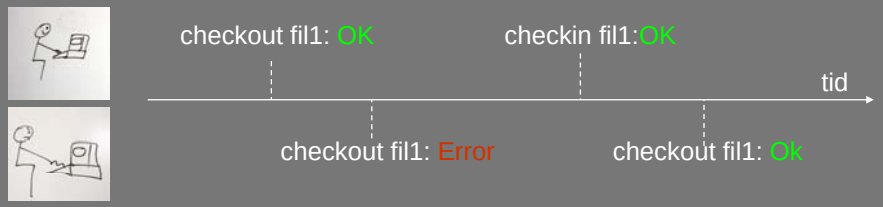


- Produktnivå merge
  Eks: Merge alle Bug-forgreninger inn i Mainline

  Her varierer verktøystøtten

Pessimistisk vs. optimistisk håndtering av parallellitet

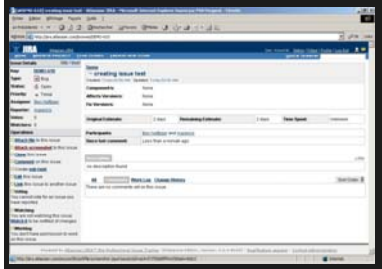
Pessimistisk (SCCS, Clearcase…)



Optimistisk (CVS, Subversion)

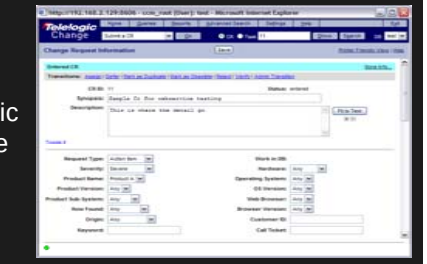
Verktøy for endringshåndtering

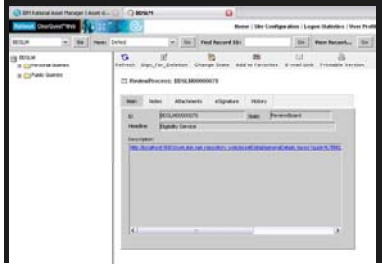
Verktøy gjør det enklere å holde oversikten over endringsforespørsler



jira

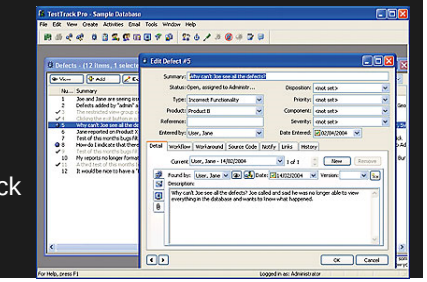
telelogic change





clearquest

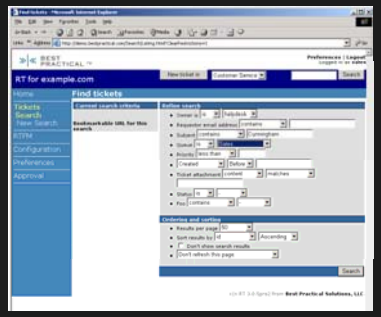
testtrack

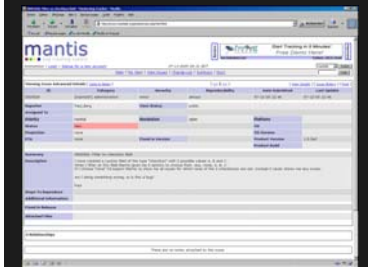


Mange gode open-source alternativer finnes

bugzilla

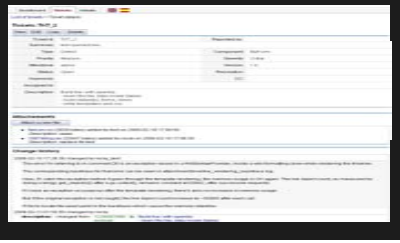
RT



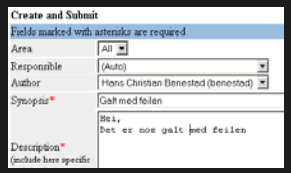


mantis

trac



Innmelding av endringer bør ha en lav terskel

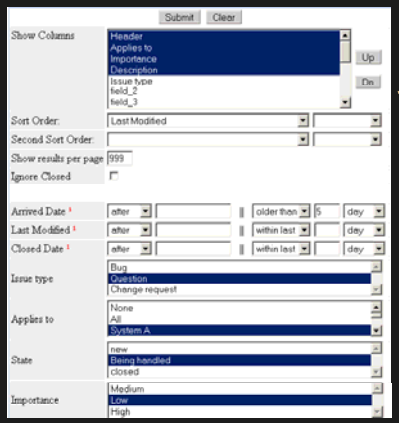


Innmelding direkte i verktøy gir best struktur

Innmelding via e-mail har lavere terskel for mange brukere

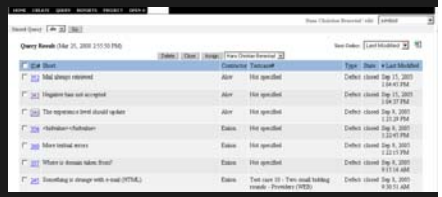
Eventuelt med direkte link fra programvaren

”Mine endringer” og fleksible søk gir god oversikt



Søkekriterier

Resultat



Integrasjon med versjonskontrollsystem er nyttig

Utvikler utfører en endring, og sjekker inn kode:

>> CVS commit –m ”ID=1 State=KlarTilSystemTest Feil bruk av peker”

CVS oppdaterer endringsdatabase

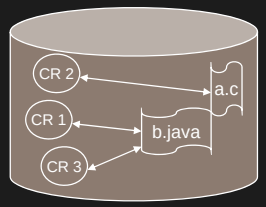
Audit Trail (Change History):

Version#2	Author: Hans Christian Benestad, Date: May 6, 2008 11:09:42 AM State: KlarTilSystemtest, Responsible: Hans Christian Benestad
Response	Feil bruk av peker
Version#1	Author: Hans Christian Benestad, Date: May 6, 2008 11:00:52 AM State: new, Responsible: Hans Christian Benestad
Description	Fillern Programmer tryner når jeg tryldker Lagre

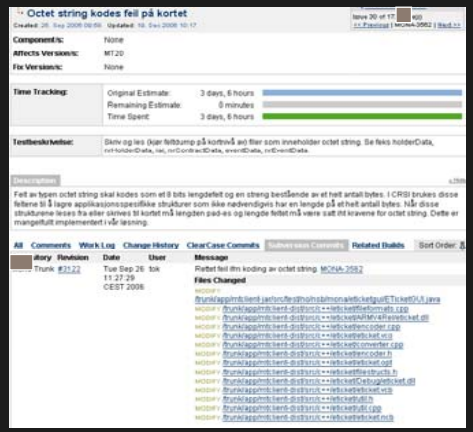
+ Enkelt for utvikler

+ Ta vare på sammenhenger mellom logiske endringer og kodeendringer

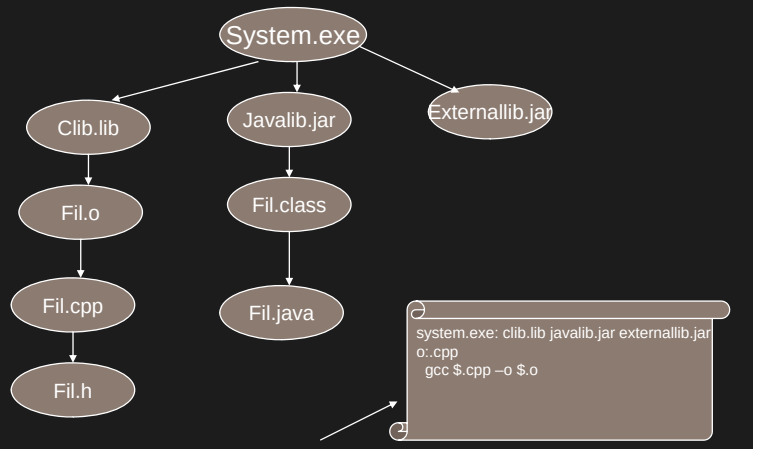
Nyttig å vise sammenhenger mellom logiske endringer og kodeendringer



CR = Change Request

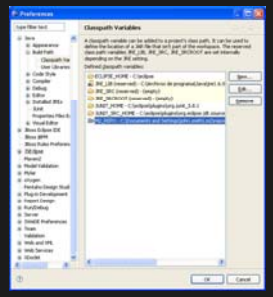


Bygging: Store programsystemer består av komponenter som utgjør en avhengighetsgraf



Byggeverktøy hjelper til med å regenerere komponenter ved behov

Byggeregler bør være sentralt definert

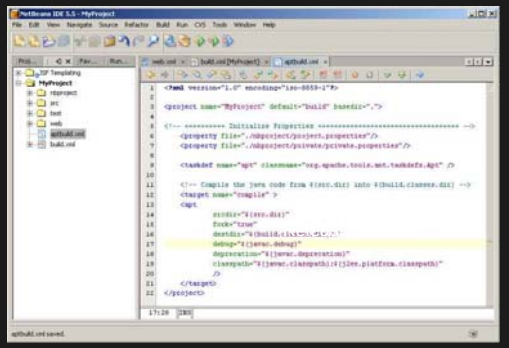


IDE'er som Eclipse oppmuntrer til å definere lokale byggeregler

Dette skalerer dårlig for større prosjekter

Byggeregler kan defineres i byggeverktøyenes beskrivelsesfiler

- Kjente verktøy: Make, Ant, Maven
- Beskrivelsesfiler spesifiserer
  - Hvilke filer inngår
  - Kompilerings- og linkeopsjoner
  - Versjoner av biblioteker og verktøy

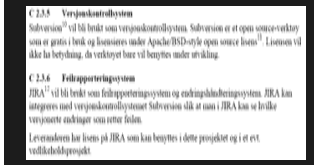


Lurt å la beskrivelsesfiler i seg selv være underlagt konfigurasjonsstyring

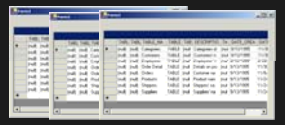
Fullt ut repeterbar bygging kan være krevende

- Krever i prinsippet konfigurasjonsstyring av
  - Kildefiler
  - Dokumentasjon
  - Eksterne/interne biblioteker
  - Generatorverkøy, som kompilatorer
  - Byggeregler

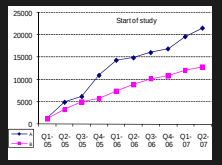
Avansert emner



Hvem skal eie SCCM-systemene?



Versjonsstyring av data



Trendmålinger basert på SCCM-data

Hvem definerer og eier SCCM-systemene?

C 2.3.5 Versjonskontrollsystem

Subversion[10] vil bli brukt som versjonskontrollsystem. Subversion er et open source-verktøy som er gratis i bruk og lisensieres under Apache/BSD-style open source lisens[11]. Lisensen vil ikke ha betydning, da verktøyet bare vil benyttes under utvikling.

C 2.3.6 Feilrapporteringssystem

JIRA[12] vil bli brukt som feilrapporteringssystem og endringshåndteringssystem. JIRA kan integreres med versjonskontrollsystemet Subversion slik at man i JIRA kan se hvilke versjonerte endringer som retter feilen.

Leverandøren har lisens på JIRA som kan benyttes i dette prosjektet og i et evt. vedlikeholdsprosjekt.

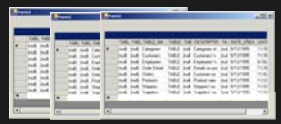
Fra den utfylte PS2000-kontrakten

Normalt vil utviklingsprosjektet styre versjonskontrollsystemet, men kunde kan og bør sette krav (se over)

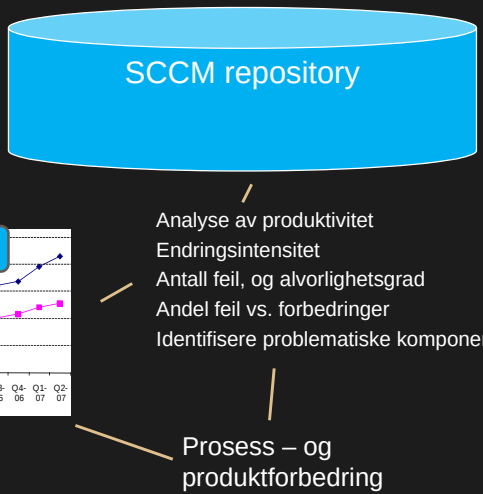
Det ligger makt i å eie endringshåndteringssystemet!

Versjonsstyring av data

- Kundens data er hellig
- Nye versjoner må ta hensyn til eksisterende datastruktur (og semantikk i data)
- I kompliserte tilfeller kan konvertering av data kreve store ressurser



Versjonsdatabaser inneholder verdifull informasjon for prosjektevaluering



Oppsummering

- God SCCM gjør livet enklere for prosjektet og utviklerne
- Valg av prosedyrer og verktøy er avhengig av behovene
- Riktig dosert så understøtter versjons- og konfigurasjonsstyring effektiv utvikling og kvalitet i leveranser

Spørsmål?