

Seriøs java med open source biblioteker

Verktøy for å håndtere kompleksiteten, og noen feller du bør ligge unna

Eirik Maus

Norsk Regnesentral

Geilo
2005-04-11

Innhold

- ▶ Problembeskrivelse
 - Eksempel: trails
- ▶ Begrepsetablering
 - Aktivitetsteori, ZPD, lean production
- ▶ Problembeskrivelse 2
 - Eksempel 2: projectlink
- ▶ Prosesser og støtte
- ▶ Verktøy (og feller)
- ▶ Feller (og verktøy)

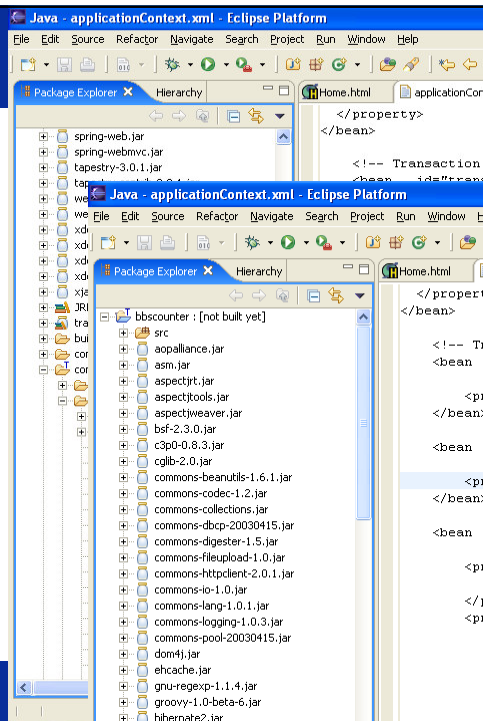
Webrammeverk?

- ▶ Ikke så få (76)
- ▶ Færre enn "class libraries" (478)
- ▶ Men bruker sikkert de fleste libraries til sammen



Problemet?

- ▶ Sånn så det ut da jeg fikk "auto-generert" et prosjektoppsett med "trails"
 - 61 jar-filer
 - 18,2 mb
- ▶ Hvordan vil du starte her?
- ▶ Hva gjør denne applikasjonen tror du?



Tema (1/2): kompleksiteten ...

- ▶ Kompleksitet
 - Moduler, avhengigheter og deres avhengigheter
 - Hvilke moduler bruker vi?
 - Og hvilke bruker de igjen
 - Funksjonalitet: hva gjør egentlig programmet?
 - Hva gjør XYZlib-1.2.jar og hvorfor er den med?
 - Konfigurasjon: hva med alle konfigurasjonsfilene?
 - Her var det mange xml-filer, gitt. Hvordan henger de sammen? Hva gjør applicationContext.xml? Db.hbm.xml?
 - Fremgangsmåte:
 - Mye av dette er verktøy, men hva gjør de?
Og hvordan skal de brukes, når og i hvilken rekkefølge?



Norsk
Regnesentral
NORWEGIAN COMPUTING CENTER

www.nr.no

Tema (2/2): ...og hvordan håndtere den?

- ▶ Tja? Noen forslag?
 - (en av premissene for foredraget er visst at det ikke skal koste en krone)



Norsk
Regnesentral
NORWEGIAN COMPUTING CENTER

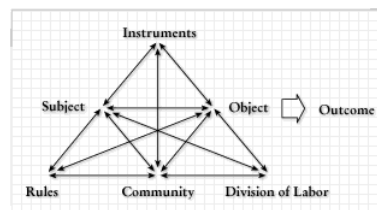
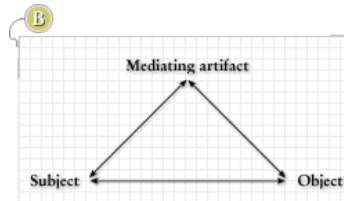
www.nr.no

Oversikt?

- ▶ Ingen har oversikt over hva som skjer i programmet
- ▶ Ingen enkeltutikler har oversikt over
 - Hele programmet
 - Alle verktøyene
 - Hvordan bygge og pakke alle deler av systemet
 - Kjøreomgivelsene
- ▶ Et kognitivt problem?

Aktiviteteteorien

- ▶ Handling skaper bevissheten
 - Via sansing og tolkning i situasjonen
 - Rettet mot objekter
 - Mediert av verktøy/ting
 - Foregår i en kulturell sammenheng
 - Danner begreper
 - Som er relative til et system av menneskelig aktivitet
 - Også mental aktivitet



Hegel og Marx : handling og tanke

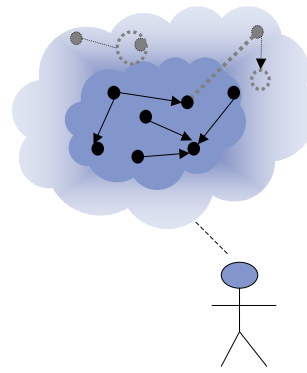
- ▶ Hegel: "virkelighet" = sammenhengende bilde av verden
så: erfaring fra det absolutte => diskrepans
men: fornuften => betrakter selvet utenfra
=> ny "virkelighet"
- ▶ Også kultur og samfunn: tilpasses "det absolutte"
- ▶ Marx: for noe tull! Menneskets arbeid og virke i felleskapet må være utgangspunktet!
 - Arbeider, skaper og skaffer seg verdier, sanser, handler og tenker i det bestående samfunnet
 - Men skaper også endring:
ny teknologi, nye arbeidsmåter,
nye klasser => sosiale strukturer og normer.

Vygotski:

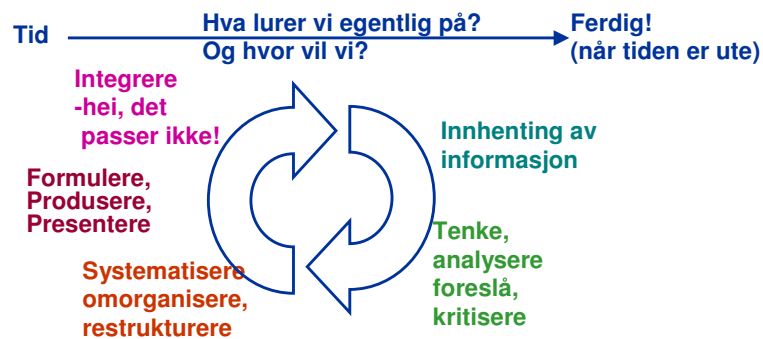
- ▶ Kunnskap = evne til å få til, læres i arbeid
- ▶ Barn
 - Vil delta med i "de stores" felleskap
 - Egen aktivitet sammen med dem
 - Imiterer og deltar i handlinger
 - Lærer ord, begreper som verktøy i dette
- ▶ Begreper, symboler: eksterne verktøy
 - For arbeid med "indre natur"
 - Lærer å arbeide uten disse etter hvert

Vygotski : ZPD & "scaffolding"

- ▶ Zone of Proximal Development
 - Sone av (full) kontroll
 - Nærhetssone
 - Utenfor rekkevidde
- ▶ Læring kan bare foregå i utkanten av kjente begreper
- ▶ Kan få til ting i utkanten med støtte, ledende spørsmål osv fra "læreren"
 - "scaffolding" (stillasbygging)



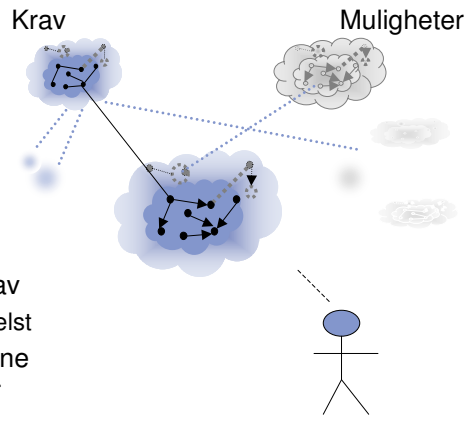
ZPD-Læring i prosjektarbeid (i grunnskolen)



- ▶ Iterasjoner, skyve grenser
 - For hver iterasjon flyttes grensen mellom hva du kan og hva du ikke kan littegrann
 - Tyngdepunkt til høyre først, så nede, så til venstre

Problem-tegning

- ▶ **Problemet:** forbinde begrepene
 - Henger ikke sammen
 - Passer ikke sammen
 - Vet lite om mulighetsområdet
 - Annet enn at det stiller krav
 - Kan ikke lage hva som helst
 - Vi må oppdage mulighetene og begrensningene under veis



Produksjon versus utvikling

- | | |
|---|---|
| <ul style="list-style-type: none">▶ Utvikling<ul style="list-style-type: none">▪ Finner oppskriften▪ Kvalitet = brukbarhet i situasjonen▪ Forskjellige resultater hver gang er bra▪ Å prøve en gang til genererer verdi | <ul style="list-style-type: none">▶ Produksjon<ul style="list-style-type: none">▪ Lager matretten▪ Kvalitet = oppfylle kravene / oppskriften▪ Variable resultater er svært uheldig▪ Å gjøre det en gang til er (skyldes) bortkastet/ ødelagt arbeid |
|---|---|

Kilde: "lean SW development" Tom & Mary Poppendieck

Prosjektarbeidet

- ▶ For dyrt å lage alt fra scratch
- ▶ Vårt bilde av "virkeligheten" (mulighetene) vil hele tiden endre seg
- ▶ Prosjektet er
 - En kognitiv aktivitet: må lære seg mye
 - En kollaborativ oppdagelsesreise i muligheten
 - Med kontinuerlig behov for ledelse
 - Hvor ikke all vil (behøve å) kunne alt

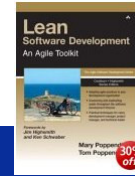
 - (ikke helt uvanlig å endre kravene heller....)
 - (minner om L97, læreplan for grunnskolen)
- ▶ "Virkeligheten" er i kontinuerlig endring

Hva vi ser etter er altså

- ▶ Arbeidsprosess
 - Inkrementell, iterativ fremgang
 - Fokus: prøving, feiling, forbedring
- ▶ Verktøy
 - Scaffolding av prosjekt-teamet:
 - Scaffolding av den enkelte i teamet
- ▶ Scaffolding =
 - Lære riktige arbeidsvaner, metoder (inngå i riktig praksis-fellesskap)
 - Få til noe på utsiden av full-kontroll-sonen, noe du ikke egentlig kan

Poppendiecks : lean (sw) developmt

- ▶ Eliminate waste
 - See waste, value stream, what adds value?
- ▶ Amplify learning
 - Feedback, iterations, synchronization
- ▶ Keep options open, decide late
 - Develop all options
 - share partially complete info,
 - decide at latest resposible point
- ▶ Empower the team
 - Self-determination, motivation, pull-based task assignment, leadership, expertise
- ▶ Build integrity in
 - Testing, refactoring, measurements, manage continuous change
- ▶ See the whole



Open source java

- ▶ Verktøy, pakker og rammeverk
 - 100-vis av hver
 - Alle problemer og alle rammeverk har et verktøy
- ▶ Det vi søker: meta-verktøy-rammeverk
 - Prosess-integrasjon av
 - Bilbliotek-bruk
 - verktøy-bruk
 - fremgangsmåter
 - Scaffolding + eliminate waste
- ▶ ANT
- ▶ MAVEN
- ▶ En skikkelig IDE

Oppgaver versus "faser"

	Design	Koding: "skape"	Bygge- "verktøy"	Sjekk status	Lede videre
Scaffold	Gode forslag	Foreslå kode- tekst	Hjelpe til å bruke verktøy du ikke kjenner		
Eliminate Waste		La være å lage det samme igjen	Unngå venting, rot. automatisering	"dynamisk oversikter"	
Learning	Deltakelse, jobbe i grupper	Par- programmering	Erfaring	Gode oversikter, debriefing	Erfaring, debriefing
Team Empower- ment				Dynamiske Oppgavelister	Fremgang gir motivasjon
Integrity	Patterns? Refactoring	Refactoring	Automatisk testing	Testing, oppgavelister	

Oversikt over verktøy

	Design	Koding	bygge	doku- mentere	Prioritere, refokusere
Scaffolding	(Google?)	code- completion, IDE-plugins	Ant, Maven	Javadoc, maven-plugins	Bugzilla, jira?
Waste- eliminaiton	Patterns	IDE: Syntax- coloring, IDE- plugins	Ant, maven, X-doclet	Maven- rapport- plugins, jira	Xplanner, jira
Læring,		IDE-tips	Cruise control?		Xplanner?
Empower- ment				Jira	
Integrity		Refactoring	Junit, HttpUnit, Jmock, Cactus Cruise control, Gump	Jira?	

Feller

- ▶ Jar hell
- ▶ XML hell
- ▶ WSDL hell
- ▶ GPL (og lisens-) hell

- ▶ Og dessuten er det vanskelig å skrive kode mot ukjente APIer

Skrive kode, debugge

- ▶ Her er ikke open source state of the art,
 - IntelliJ IDEA ruler bigtime
 - Men eclipse (og netbeans 4.1) har fine plugins
- ▶ Feller og verktøy: for mange til å nevne

- ▶ Scaffolding, waste, learning.. (alle kategorier):
 - IDE med "intellisense" pop-up på parametere o.l.
- ▶ Scaffold: foreslå kode, antyde "code smells"
- ▶ Waste-elim: unngå gå skrive feil: "rød streker", auto-import, søkefunksjoner,... plugins, særlig for byggeverktøyet!
- ▶ Integrity: REFACTORING-støtte!!!,

Bygging, pakking,.....: feller

- ▶ Ha alle jar-filer for å compilere: hvor kommer de fra?
- ▶ Ikke bare kompliere: jar, war, xml-descriptorer, metadata, mapping-filer, digital signering, deployment-descriptorer, auto-generert kode,
 - Selv å starte for å teste er vanskelig
- ▶ Scaffold, waste-elimination, learning (by doing), integrity,....
 - Byggescrypt med ANT!
(eller maven)

Hva er Apache Ant

- ▶ ANT: Byggesystem laget i java (for java?)
- ▶ "Make without make's wrinkles"
- ▶ Automatisering av bygging
 - ...og alle trinnvise oppgaver basert på filer og tekster
- ▶ XML-fil beskriver hva ANT skal gjøre
- ▶ Deklarativ, men med imperativer i oppgavene

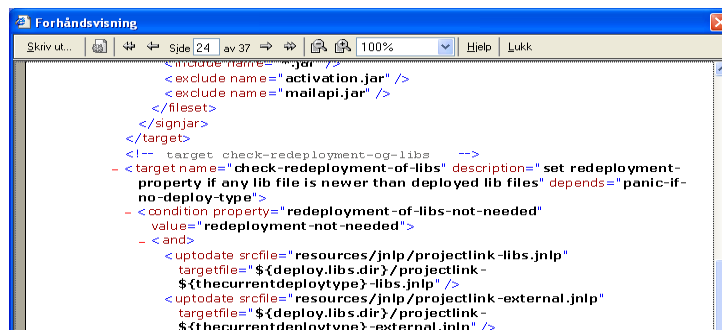
- ▶ <http://ant.apache.org>

Ant: god bruk

- ▶ "scaffold hverandre"
 - Task for nedlasting av nødv. biblioteker/verktøy
 - De andre har ikke lært det du har lært i dag
 - Skjønner/kjenner ikke de verktøy du har tatt i bruk
 - Vet ikke hvor du har det fra
- ▶ "waste-eliminate hverandre"
 - Skal virke på andres maskiner hvor ting er installert i andre kataloger UTEN at de endrer noe
 - Bruk `fil.properties`

ANT feller

- ▶ Lang og uoversiktlig fil
- ▶ Kan nesten aldri brukes om igjen
 - Skrives på nytt i hvert prosjekt: waste, så det holder



```
<include name="*.jar" />
<exclude name="activation.jar" />
<exclude name="mailapi.jar" />
</fileset>
</signjar>
</target>
<!-- target check-redeployment-og-libs -->
- <target name="check-redeployment-of-libs" description="set redeployment-
property if any lib file is newer than deployed lib files" depends="panic-if-
no-deploy-type">
- <condition property="redeployment-of-libs-not-needed"
value="redeployment-not-needed">
- <and>
<update srcfile="resources/jnlp/projectlink-libs.jnlp"
targetfile="${deploy.libs.dir}/projectlink-
${thecurrentdeploytype}-libs.jnlp" />
<update srcfile="resources/jnlp/projectlink-external.jnlp"
targetfile="${deploy.libs.dir}/projectlink-
${thecurrentdeploytype}-external.inln" />
```

Hva er maven

- ▶ Super-make for superkompliserte bygginger
- ▶ Gjenbruk av script (ikke som i ant)
- ▶ Scriptsamling = akkumulasjon av erfaring
- ▶ Predefinert prosjektstruktur:
 - sånn gjør du det best
 - (stol på oss, vi har gjort det før og det funker)

Maven: "ant without ant's wrinkles"

- ▶ Byggesystem, basert på plugins
- ▶ Prosjekt:
 - Lager KUN EN artifakt (jar, war) per prosjekt
 - Men støtter hierarkiske prosjekter, avhengigheter
 - POM: project object model: definerer HVA
 - Type artefakt
 - "fast" katalogstruktur
 - Avhengigheter: biblioteker (lastes ned automatisk)
 - Rapporter: javadoc, junit-tests, ,....(ørlen)
 - Script: definerer hvordan :
 - preGoal="java:compile" attainGoal="antlr:grammar"
- ▶ Alt laget i plugins, som benyttes i alle prosjekter
 - Hauger av plugins i sentralt repository

Hva består maven av?

- ▶ "Maven" kjerne
 - Avhengighetsgraf-system for "goals" du vil bygge
 - Jelly: xml-script-motor
 - Leser prosjekt-descriptor : data
 - Laster ned alle avhengigheter fra internet repositories !
- ▶ Maven plugins : script for ulike oppgaver
 - Ca 90 "maven plugins" + 50 uavhenige
 - Jar plugin, java plugin, javadoc, struts, latex, xdoc dokumentasjon => websider, Junit test-rapport, webserver (!), FAQ, Hibernate, JBoss,
- ▶ Prosjektet ditt
 - Project.xml : data om prosjektet: navn, type, filer, kataloger
 - Maven.xml : spesielle regler for bygging

Project.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project>
  <pomVersion>3</pomVersion>
  <name>NR Space 2</name>
  <id>nr-space2</id>
  <currentVersion>0.1-SNAPSHOT</currentVersion>
  <url>http://intern.nr.no/java/nr-space2/${pom.art
  <organization>
    <name>NR</name>
    <url>http://intern.nr.no/java/nr-space2</url>
    <logo>http://www.nr.no/images/common/NRlogo_web
  </organization>

  <inceptionYear>2004</inceptionYear>
  <package>no.nr.space2</package>
  <logo>http://www.nr.no/images/common/NRlogo_web.g
  ...

  <distributionDirectory>
    /I:/java/space2/${pom.artifactId}/
  </distributionDirectory>

  <repository>
    ....
  </repository>

  <dependencies>
    <dependency>
      <id>junit</id>
      <version>3.8.1</version>
    </dependency>
    ...
  </dependencies>

  <build>
    <sourceDirectory>
      ${basedir}/src/main/java
    </sourceDirectory>
    ....
  </build>
</project>
```


Maven.xml (hele)

```
<project
  default="java:jar"
  xmlns:j="jelly:core"
  xmlns:maven="jelly:maven"
  xmlns:deploy="deploy"
  xmlns:a="jelly:ant"
  xmlns:u="jelly:util"
  >

  <preGoal name="java:compile">
    <attainGoal name="run-jjtree-on-files-if-needed" />
  </preGoal>

  <goal name="run-jjtree-on-files-if-needed">
    <attainGoal name="javacc:jjtree-generate" />
    <attainGoal name="javacc:javacc-generate" />

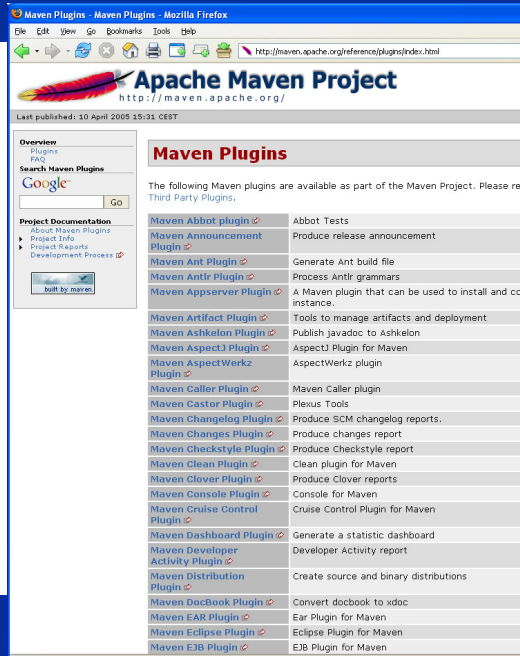
    <a:path id="maven.javacc.compile.src.set"
      location="{maven.build.dir}/generated-src/main/java"/>
    <maven:addPath id="maven.compile.src.set"
      refid="maven.javacc.compile.src.set"/>
  </goal>
</project>
```

Maven plugin

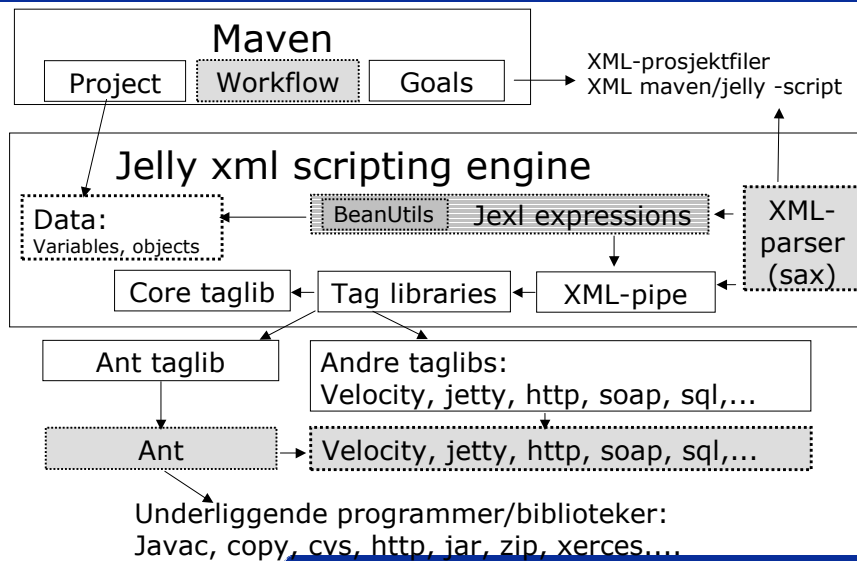
- ▶ Project.xml: avhengigheter, ressurser
- ▶ Plugin.properties
- ▶ Project.properties
- ▶ Plugin.jelly : selve scriptene, som maven.xml
- ▶ (evt. Java class/jar-filer)

Maven plugins

- ▶ Flere repositories,
- ▶ Med plugins,
- ▶ Som har goals,..



Maven (og Jelly) teknologioversikt



Keep options open, decide late WSDL hell

- ▶ Vi måtte bytte Web-service bibliotek
 - Bug, Lisens-tull
- ▶ Lettere sagt enn gjort:
 - Ikke kompatible Server Interface => WSDL
 - Ikke kompatible WSDL => klient kildekode
- ▶ Løsning: løsning?
 - Unngå sammensatte datatyper!
 - Send og motta String: (som inneholder XML)
 - Men hva er da poenget med web services?

Keep options open, decide late GPL hell

- ▶ Vi måtte også bytte Database midtveis
 - JDBC-driver plutselig GPL :
Krever at hele applikasjonen GPL
 - Men: hva med alle de andre bibliotekene?
- ▶ Løsning:
 - Skygg unna GPL
 - Vær forsiktig
- ▶ Problemet var vel at vi valgte for tidlig.
- ▶ Nå bruker vi Hibernate

slutt