Kan vi skape mye mere verdi i softwareprosjekter?

Softwarebransjen har i den senere tid vært fokusert på å øke produktiviteten. Vi tar stadig i bruk ny teknologi og moderne systemutviklingsmetoder for å optimalisere produktiviteten i gjennomføringen av softwareutvikling.

Likevel så stanger vi hodet i veggen dersom vi betrakter mange softwareprosjekter fra et "verdiskapning over livsløpet" perspektiv. Dette foredraget vil sette fingeren på to hovedelementer som kan være med på å endre dette bildet.

Disclaimer:

Min erfaring og virkelighetsfokus er sentrert rundt skreddersøm og enterprise systemer. Hvor relevant deler av disse utfordringene er i andre biter av softwarebransjen vil variere.

Innledende postulater

- Vi har mistet kontrollen på valgene vi gjør i programvareutvikling
- Vi mangler gode byggeklosser som skal til for at vi skal kunne ta bedre valg
- Vi er derfor effektivt hindret i å skape mye større verdi i programvareutvikling

Agenda

Introduksjon

- **Hva er problemet?**
- **Hva er status?**

Hvordan tas valg?

Arkitektur

Oppsummering og konklusjoner

Hvem er Totto?

- Webstep - konsulent
- Cantara
- Sun Java Champion
- Advisory Board, java.net

- ex- JavaZone og javaBin sjef
- 30 år siden jeg begynte å få betalt for å skrive kode...
  - 10 år som applikajsonsutvikler
  - 20 år som systemutvikler

Intro – Status idag

- *”By adding control to a process, the cost increases – always!!”*

Intro – Hva er problemet?

- **Produktivitet**
  - Vi har fått produktive utviklere
  - Men vi lager ”**feil**” løsninger
  - Og vi har MYE waste
  - Og løsningene har for kort levetid
  - Og eskalerende forvaltningskostnader

- **Gjenbruksmantraet**…
- **Less for less-movement?**
- **Eller…?**

Intro – Hva er status?

- **Agile/knowledge world**
- Norge som eksempel)
  - Høy produktivitet blant utviklere
  - Gode til å håndtere endringer i prosjekt
  - Dårlig på forutsigbarhet
  - Løsningene tåler i liten grad tidens tann
  - Dyrt forvaltede løsninger, høy rework faktor

- **Prosess & Software engineering world**
- (India som eksempel)
  - Gjevn produktivitet
  - Høy grad av standariseirng (rammeveerk og patterns)
  - Bra forutsigbarhet (estimater og systemets egenskaper)
  - Løsningene tåler tidens tann rimelig bra
  - Linærre forvaltningskostnader

www.cantara.no

Del 1. Valg. Tar vi de riktige valgene?

Vi begynner med å se på prosessene og resultatene rundt nøkkelbeslutninger for implementasjon, hvor vi eksemplifiserer hvor tilfeldig viktige valg faktisk blir tatt. Vi vil også se på hvordan et fokus på teknologi-egenskaper kan gi oss et rammeverk for å ta valg som kan gi betydelig større verdi i softwareprosjekter.

CANTARA

www.cantara.no

Hvordan tas valg, egentlig?

- Myter
- Corporate policies
- People and processes
- Valgets kvaler – for mange alternativer

There can

Hvordan tas valg - Myter

It is a myth!! - Example: EJB

- 1997 - superhot! All vendors(-Microsoft), all hotshots super stoked!
- 1998 - Pet store. Statefull FUD. Year of EJB benchmarking.
- 2000 - Hotshots turn against EJB. No silver bullet (again)
- 2002 - Year of Spring in early adopters markets (Norway)
- 2005 - Hotshots turn agains Spring and Java web development. Look at RAILS!!
- 2009 - Hotshots turn agains Rails. Look at Scala Lift.

There are plenty of examples:

- RDBMS, Web applications, .Net, Windows, REST,
- SOA and Web Services,, Cloud, [.add your favourite one..]

Hvordan tas valg - Myter

Things to remember:

- There is no silver bullet.
- Use the brain, Developer!
- Fun to follow the hype - but it does not create value
- Different problems require different solutions

(like in the rest of the world)

Hvordan tas valg – Corporate Policies

Thou shalt only use Java/J2EE/.NET/LAMP[...]!

• Reduces the value created from technology
• Increases the startup and training costs
• You create a **VERY big hammer** that must be used for ALL tasks: hitting nails and changing lightbulbs.

Special considerations regarding training costs.
• Company policies increase them!
• Using inadequate tools create solutions that are hard to understand. Training debt!
• When problems seems hard to solve
  • **Stop and think: Am I using the right tools for the job?**

Special considerations regarding operations.
• Is it a good idea to be dictated by your operator when it comes to finding the right tools for your job.

Hvordan tas valg - Psykologi

Fear of making wrong decisions:

- Sartre: "Med valg følger angst"
- "Nobody has ever been fired for choosing IBM"
- "You can´t go wrong with beige“

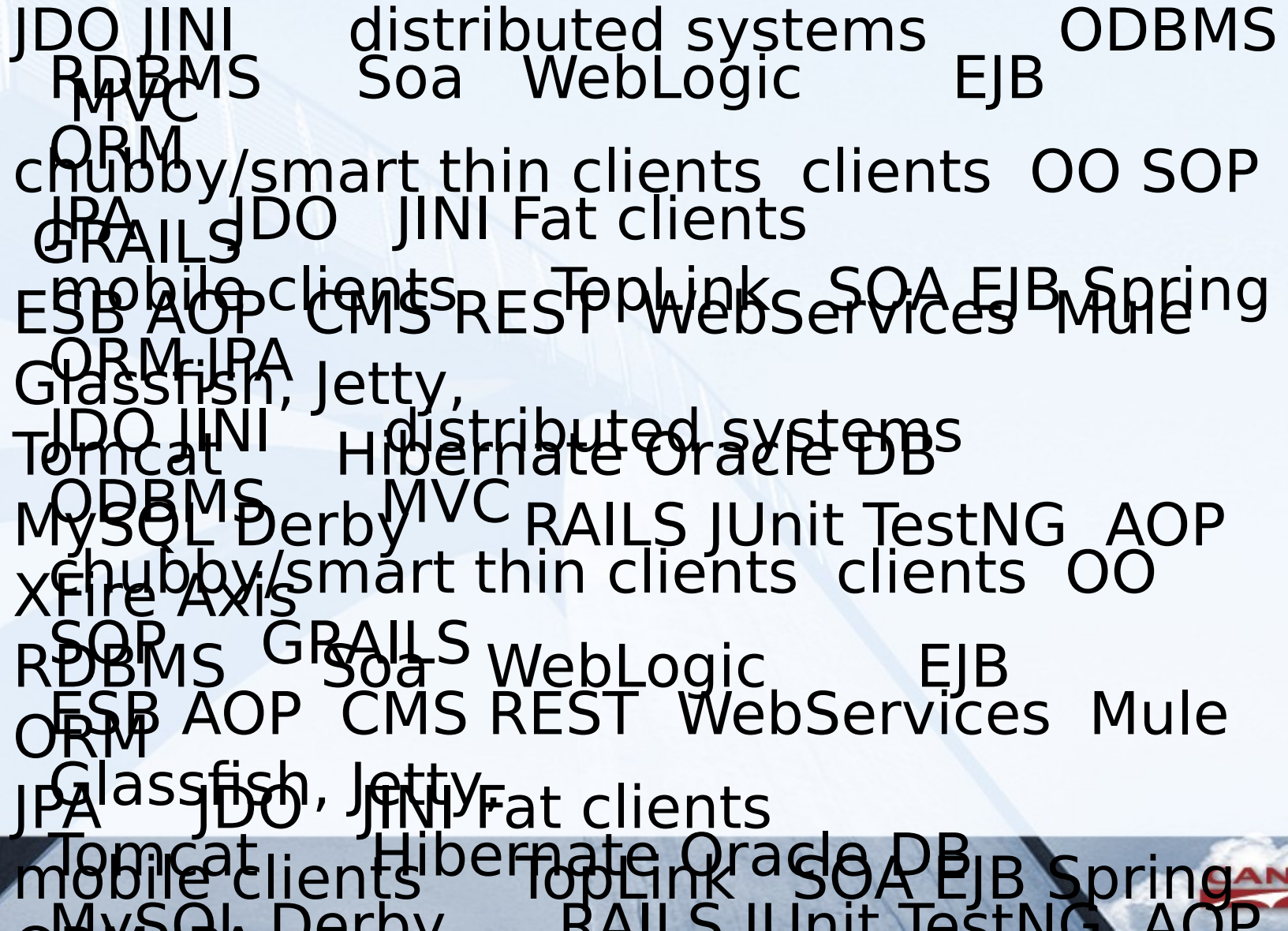
Polarization

- Lightweight vs Suites
- Linux vs Microsoft
- Remember: There can be only one!
- A religion war!
- Easier to get attention when crying out loud.
- Most effective way to create a revolution.

Hvordan tas valg – For mange alternativer

RDBMS Soa WebLogic EJB
ORM
JPA JDO JINI Fat clients
mobile clients TopLink SOA EJB Spring
ORM JPA
JDO JINI distributed systems
ODBMS MVC
chubby/smart thin clients clients OO
SOP GRAILS
ESB AOP CMS REST WebServices Mule
Glassfish, Jetty,
Tomcat Hibernate Oracle DB
MySQL Derby RAILS JUnit TestNG AOP

Hvordan tas valg – For mange alternativer



Hvordan tas valg – ta et valg..

Oppgave: Tilgjengeliggjøre informasjon på webben

Typiske alternativer:
OpenCMS, Portal frameworks, Wikis, File-based, Web Publishing systems, Document Management systems, Collaboration Systems, EzPublish, SharePoint, MediaWiki, Homegrown, Alfresco, [...] :

Vi sammenligner epler og bananer.. og ender opp med å velge enten

- alternativet med flest features
- eller produktet fra den største leverandøren...

• Og glemmer helt problemet vi skal løse **- The context.**

Hvordan tas valg – For mange alternativer

We mix implementations, technology and context.

- Publish articles to the web(Context)
- Content management system (Technology)
- OpenCMS (Implementation)

Hvordan tas valg – Andre strategier…

- Read the white papers?
- Try out demos
- Create prototypes?
- Experience, own or others?
- Implement more than one solution?
- Go with the flow?

www.cantara.no

***Sigh* .... This is hard stuff...**

Del 2. Arkitektur.
Arkitektur er en brannfakkel om dagen, og det ikke uten grunn. Det skrives opp og i mente om arkitektur og anti-arkitektur. Vi vil i denne delen av presentasjonen undersøke og sette spørsmålstegnet på om vi kanskje i 2009 begynner å se konturene av gode mulige arkitekturelle byggesteiner som faktisk er forutsetningen for å investere i arkitekturen i et system og hvordan disse kombinert med å ta bedre valg kan være en måte å skape mye mere verdi enn dagens norm i softwareprosjekter.

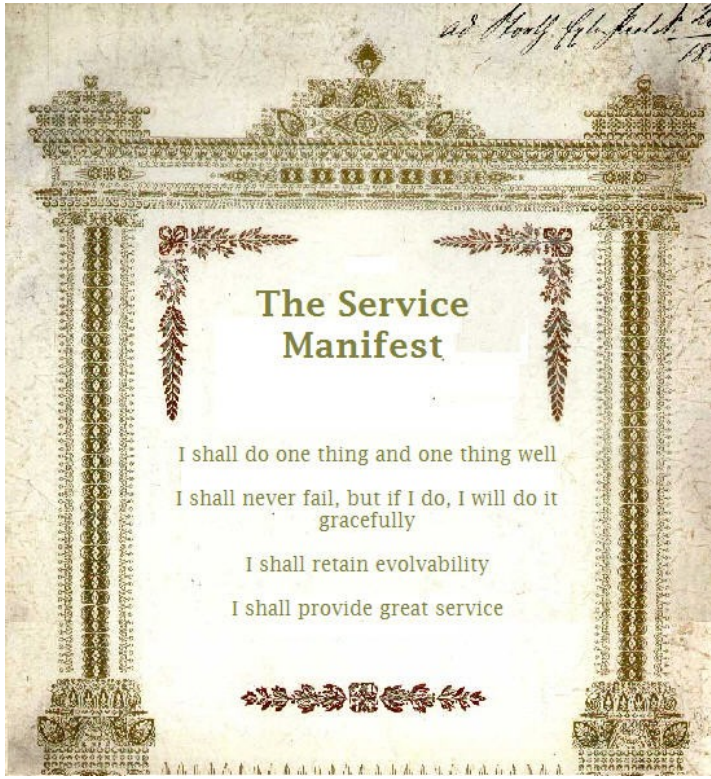
Arkitektur- for mye eller for lite?

Ting vellykede arkitekturer har til felles

- **Clear and consistent responsibilities powers all great architectures**

- **Tjenestene må gjøre en ting og en ting godt** (northbound simplicity is paramount)

- Spørsmål:
  Lever dagens arkitekturartifakter opp til disse egenskapene?



Arkitektur- Rammeverk

Mennesker har alltid ønsket seg oppskrifter. Men de fleste IT prosjektene er forskjellige, spesielt er de utsatt for mennesker, makt og styringsprinsipper. De fleste oppskriftene forenkler bort denne virkeligheten...

Arkitektur – Axiomer og prinsipper

Clear and consistant responsibillity powers all great architectures

Arkitektur – Mulige nye byggesteiner

A counter measure to faith, culture and religion.

We need some kind of help in order to resist the temptation of religion politics and other lies!

Categorization

•To make good selections, we need to categorize the problems and Contexts

•We tried to make a simple guide to this process by focusing on the matrix of contexts, solution qualities and technologies

Arkitektur – Mulige nye byggesteiner

To make good selections, we need to categorize the problems and
Contexts

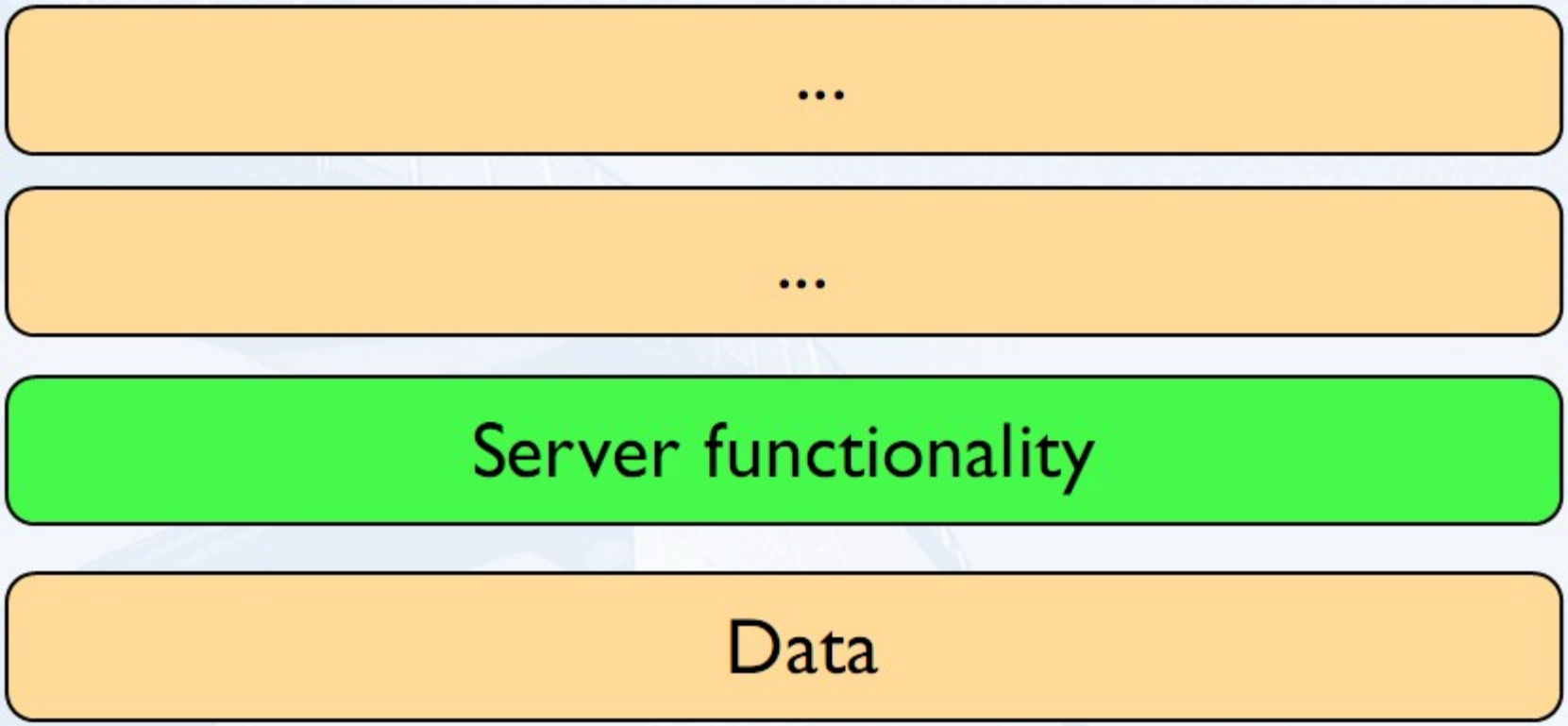
We tried to make a simple guide to this process by focusing on the
matrix of contexts, solution qualities and technologies

Remember:

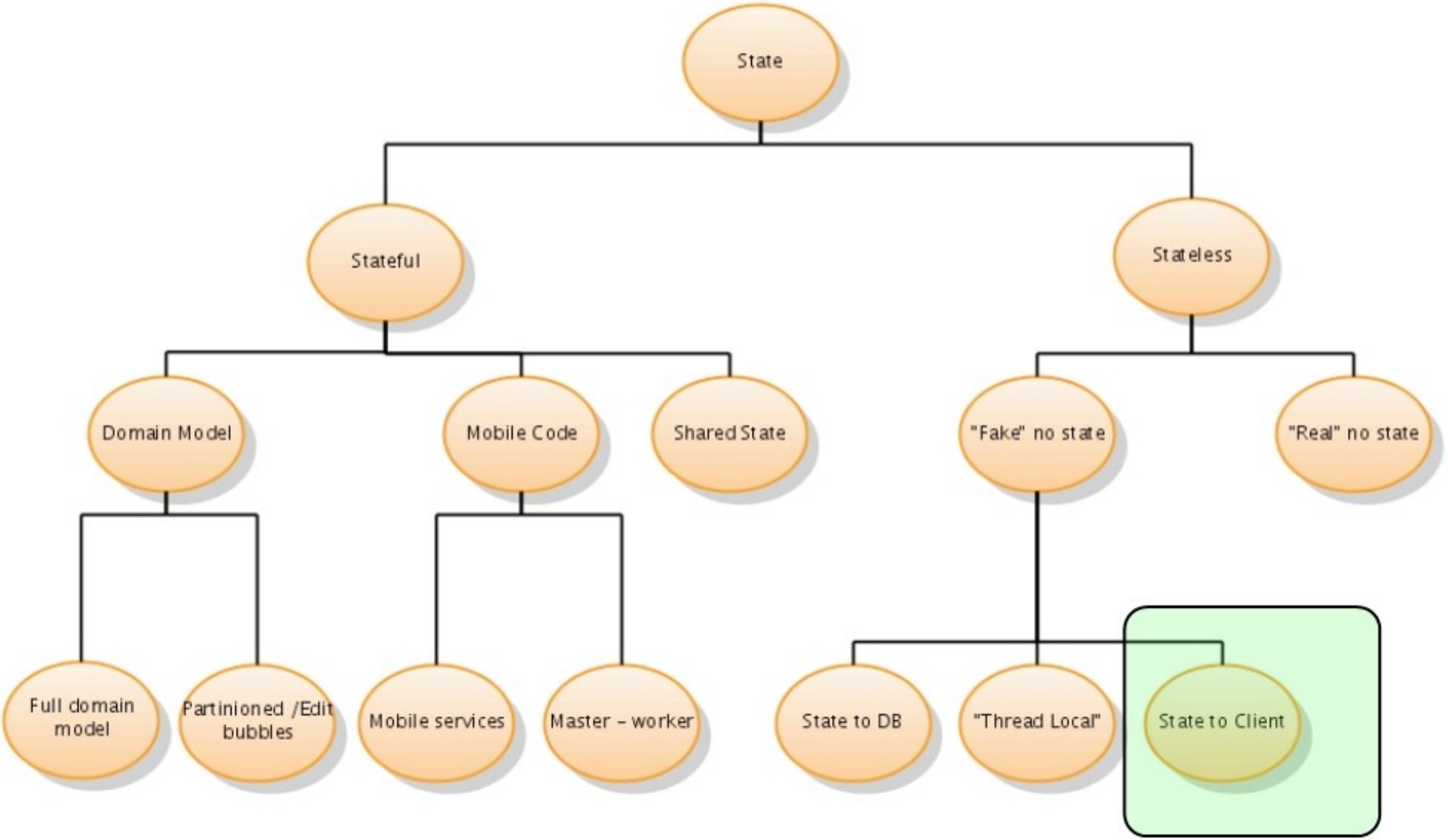
We need architectural building blocks which represent the mantra:

Do one thing and one thing well!!

Arkitektur – Server functionality



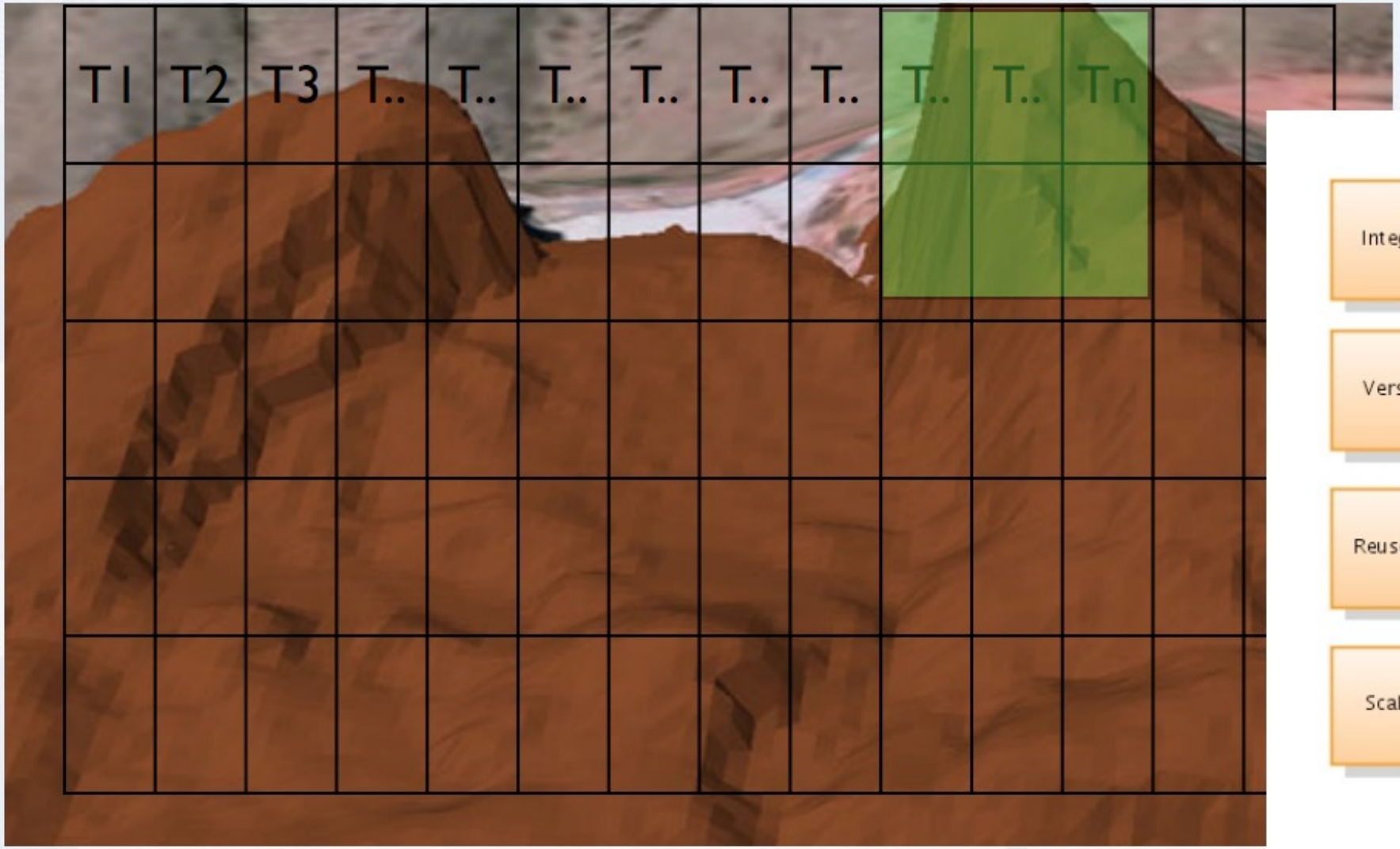
Arkitektur – Context – Server



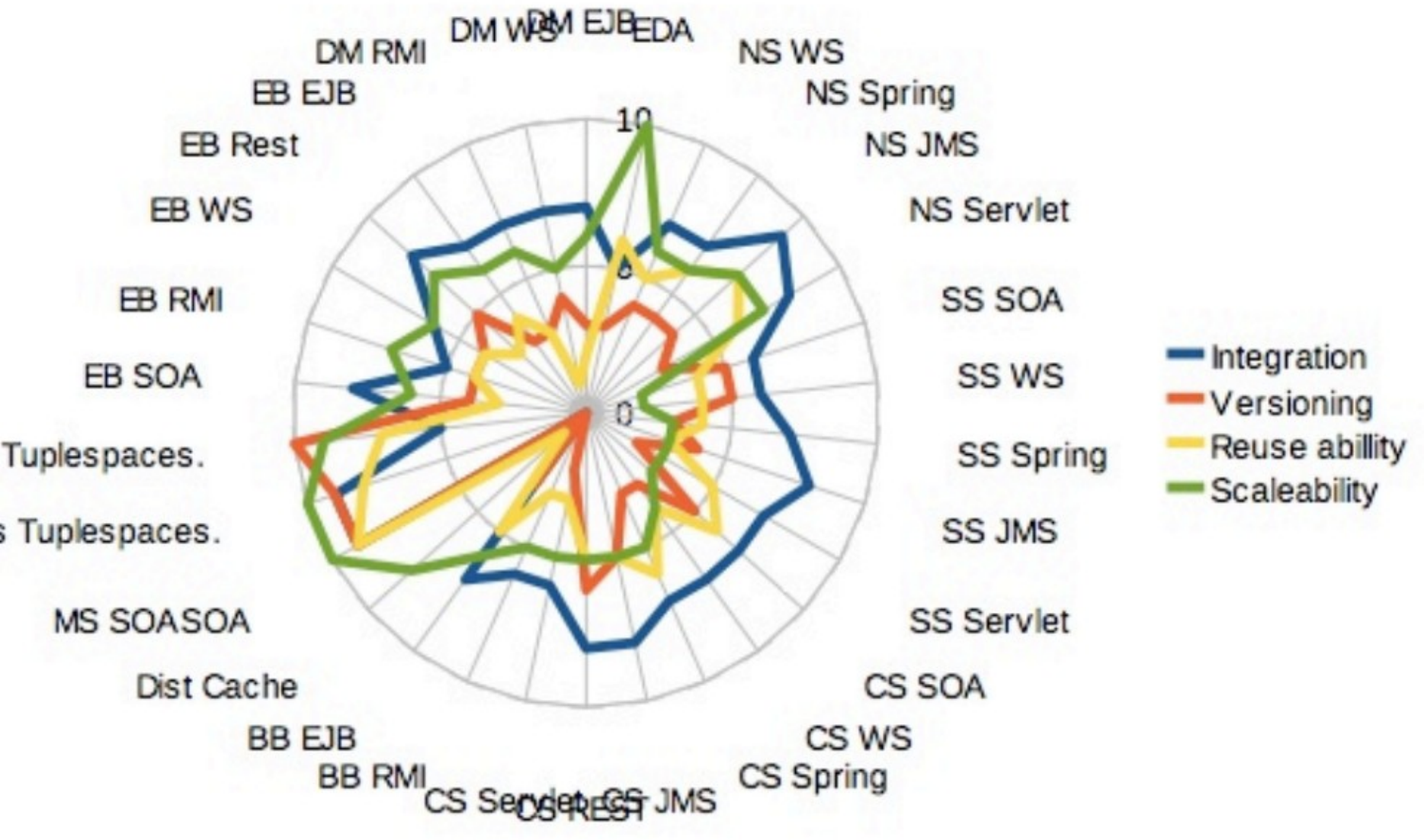
Arkitektur – Produkt versus å bygge selv

Servlets	Rest	JMS	WS	SOA	JEE Light
6	8	8	7	7	7
2	6	5	3	5	3
3	5	5	4	6	6
5	5	5	4	3	5

Visualized as landscape model



Or visualized as a Spidergraph



WTF?

Keep focus on the right things!

- Divide and conquer
- Prioritize and rate qualities

And remember technology have different strengths in different contexts

Tankeeksperiment

Web-based customer system for department in large enterprise (10k+ emp.)

- Existing database
- No suitable API´s to integrate with.
- Yet another database driven web application.

Which translates to the following qualities:

Database layer

- No though requirements in Scalability, Availability, Integrity, Query functionality, Latency, Persistence and Transactions.

Service layer

- No though requirements in Integration, Versioning, Reusability, Scalability.

Tankeeksperiment – Contekst detaljer

Java server environment.

- All developers know Java.
- Corporate policy: Java, Oracle og JPA.

Standard 3 layers webapp:

- Frontend: JSF or Struts2
- Domain logic: Spring framework
- Persistence: Hibernate ORM to DB
- **Nobody got fired from choosing IBM**

Nei, NEI, NEI!!

Tankeeksperiment – Contekst detaljer

Stop and think!

- **There are much more suitable alternatives**
- **much more fun**
- **much more productive**
- **and still open source and Java**

Let´s look at 4 key Java open source alternatives

- **Groovy on Grails**
- **JRuby on Rails**
- **Scala and Lift**
- **4GL / model driven**

Tankeeksperiment – Contekst detaljer

JRuby on Rails

- Productivity factor: Ten times as productive.
- Fun factor: Five times more fun.

Groovy on grails

- Productivity factor: Six times as productive
- Fun factor: Five times more fun..

Scala and Lift

- Productivity factor: Two times as productive.
- Fun factor: Ten times more fun. On top of developers hype curve.

4(5)GL / model driven

- Productivity factor: five to twenty times as

Tankeeksperiment – What happens next…?

Great success!!

- 80% of the department is using the app on weekly basis.
- Rumors of the application spreads like wild fire!
- Developers and stakeholders are treated like heroes.

… now everyone wants the application!

The landscape changes:

- A large number of databases and CRM systems.
- A myriad of customer structures.
- Off the shelf software
- 24/7 requirements

Tankeeksperiment – konsekvenser

Konsekvenser

- Data mastering becomes a huge issue.
- Not possible to do updates and/or de
- Need some kind of data consolidation.
- Rate of change in requirements skyrocket
- 24/7 requirements
- Our simple integration strategy is now void.

Which translates to the following qualities:

Database layer

- Though requirements in Scalability, Availability, Integrity, Latency.

Service layer

- Though requirements in Integration and Versioning.

We are no longer free to pick and choose freely.

CANTARA

Tankeeksperiment – konsekvenser

Our context has changed dramatically!

Old: "Simple web-based customer system"
New: "Enterprise customer dashboard"

Tankeeksperiment – myter

“Our Enterprise Corporate policy would have prevented this!”

- Wrong: No application, no problem!
- Wrong: Poor application, same problem!

“We need to re-implement the solution on a platform that supports this new complex scenario!”

- Wrong: There is nothing wrong with the application!
- Wrong: The new requirements can be handled on the existing platform.
- Wrong: The enterprise platform does not solve any problems.

.... og hvis vi velger å begynne på nytt, så er det uansett ikke den største investeringen man kaster…

Tankeeksperiment – myter

Enterprise corporate policy did/will not save us (this time)

- Let´s focus on the problems at hand:
- Server layer: Integration, Versioning
- Data layer: Scalability, Latency, Availability,

We can use the landscape to rate integration technologies for each individual data source.

- Keep focus on the problems at hand.
- Server component need to extend its responsibility (since we can´t control the data sources)
- caching
- domain objects
- correlation and mapping

Tankeeksperiment – myter

Servlets	Rest	JMS	WS	SOA	JEE Light
6	8	8	7	7	7
2	6	5	3	5	3
3	5	5	4	6	6
5	5	5	4	3	5

Integration

Versioning

Reuse ability

Scalability

Oppsummering og konklusjoner

Vi er blitt flinke og produktive som utviklere

- **Men vi lager feil løsninger**
- **Med feil egenskaper**
- **Siden vi har mistet kontrollen på valgene vi gjør**
- **Og produserer derfor langt**