Developers guide to server-side productivity and fun using Open Source

A flight through the landscape of feature sets, technologies and Open Source implementations

Motivation

- We want to increase productivity and quality of software!
- We want to empower developers to have more fun!
- We can fix this by making better decisions!

Motivation

ease productivity and
e!

ower developers to have

- We can fix this by making better decisions!

Motivation

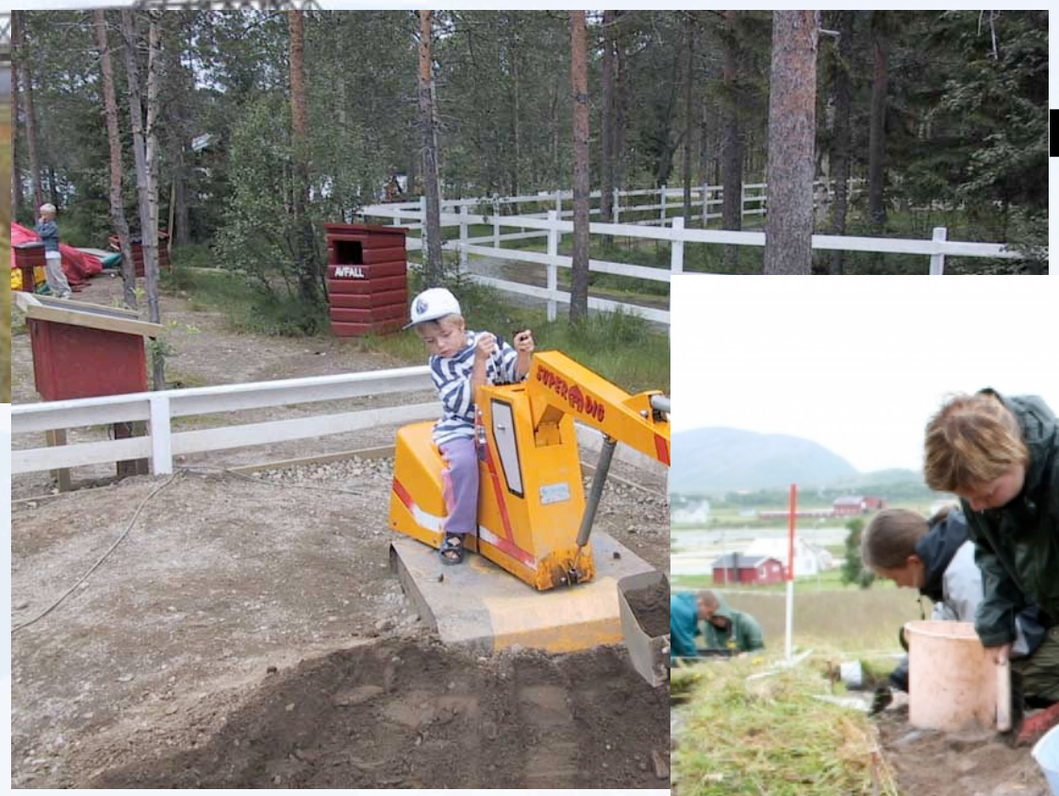
ease productivity and
e!

rs to have

- We can ~ decisions!

Motivation

ease productivity and
e!



rs to have

- We can

CANTARA

Speakers

- Tobias K Torrissen
  - javaBin, JavaZone, Know IT Objectnet, Cantara, Stiftelsen for fremme av programvareutvikling i Norge.
- Totto - Thor Henning Hetland
  - javaBin, JavaZone, Java Champion, IASA,Stiftelsen for fremme av programvareutvikling i Norge.
  - 30 years of experience as developer, Cantara Java.net community leader.

CANTARA

Agenda

- What challenges are we facing
- A brief look at potential aid
- Case study
- Conclusion
- Q&A

What challenges are we facing

Challenges

- Myths
- Too many options
- Corporate policies
- People and processes

There can only be one!

There can only be one!

CANTARA

There can only be one!

- It is a myth!! - Example: EJB

There can only be one!

- It is a myth!! - Example: EJB
  - 1997 - superhot! All vendors(-Microsoft), all hotshots super stoked!

There can only be one!

- It is a myth!! - Example: EJB
  - 1997 - superhot! All vendors(-Microsoft), all hotshots super stoked!
  - 1998 - Pet store. Statefull FUD. Year of EJB benchmarking.

There can only be one!

- 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)

There can only be one!

- 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)

There can only be one!

- 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!!

There can only be one!

- 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 and Lift.

Myth: There can only be one!

- There are plenty of examples:
  - RDBMS
  - Web applications
  - .Net
  - Windows
  - SOA and Web Services
  - REST
  - [...]

Myth: There can be only one!

- 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)

Too many options

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 XFire Axis
Spring WS Client server

Too many options

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

ESB AOP CMS REST WebServices Mule Glassfish Jetty,
RDBMS Soa WebLogic EJB ORM
Tomcat Hibernate Oracle DB
JPA JDO JINI Fat clients LINQ AOP
MySQL Derby RAILS JUnit TestNG AOP XFire Axis
mobile clients TopLink SOA EJB Spring ORM JPA
Spring WS Client server
JDO JINI distributed systems ODBMS MVC

Too many options

- We mix implementations, technology and context.
  - Publish articles to the web(Context)
  - Content management system (Technology)
  - OpenCMS (Implementation)

Too many options

Too many options

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

Too many options

- Make a choice: OpenCMS, Portal frameworks, Wikis, File-based, Web Publishing systems, Document Management systems, Collaboration Systems, EzPublish, SharePoint, MediaWiki, Homegrown, Alfresco, [...] :
- We compare apples and bananas

Too many options

- Make a choice: OpenCMS, Portal frameworks, Wikis, File-based, Web Publishing systems, Document Management systems, Collaboration Systems, EzPublish, SharePoint, MediaWiki, Homegrown, Alfresco, [...] :
- We compare apples and bananas
- We tend to select

Too many options

- Make a choice: OpenCMS, Portal frameworks, Wikis, File-based, Web Publishing systems, Document Management systems, Collaboration Systems, EzPublish, SharePoint, MediaWiki, Homegrown, Alfresco, [...] :
- We compare apples and bananas
- We tend to select
  - Most features

Too many options

- Make a choice: OpenCMS, Portal frameworks, Wikis, File-based, Web Publishing systems, Document Management systems, Collaboration Systems, EzPublish, SharePoint, MediaWiki, Homegrown, Alfresco, [...] :
- We compare apples and bananas
- We tend to select
  - Most features
  - Biggest vendor.

Too many options

- Make a choice: OpenCMS, Portal frameworks, Wikis, File-based, Web Publishing systems, Document Management systems, Collaboration Systems, EzPublish, SharePoint, MediaWiki, Homegrown, Alfresco, [...] :
- We compare apples and bananas
- We tend to select
  - Most features
  - Biggest vendor.
- We forget the problem at hand - The context.

Too many options

- Make a choice: OpenCMS, Portal frameworks, Wikis, File-based, Web Publishing systems, Document Management systems, Collaboration Systems, EzPublish, SharePoint, MediaWiki, Homegrown, Alfresco, [...] :
- We compare apples and bananas
- We tend to select
  - Most features
  - Biggest vendor.
- We forget the problem at hand - The context.
- **We have lost control!!**

Too many options

- Make a choice: OpenCMS, Portal frameworks, Wikis, File-based, Web Publishing systems, Document Management systems, Collaboration Systems, EzPublish, SharePoint, MediaWiki, Homegrown, Alfresco, [...] :
- We compare apples and bananas
- We tend to select
  - Most features
  - Biggest vendor.
- We forget the problem at hand - The context.
- **We have lost control!!**

Company 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.

Company policies

- 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?

Company policies

- 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.

People and process

- Things that influence us:
  - Psycological aspects
  - Cultural aspects
  - Corporate policies
- Is there a better way?

Psychological aspects

- 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"

Culture and religion

- 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.

Other strategies

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

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

A brief look at potential aid

CANTARA

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 and exemplifying solutions.

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

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

Toplevel "Layered" breakdown

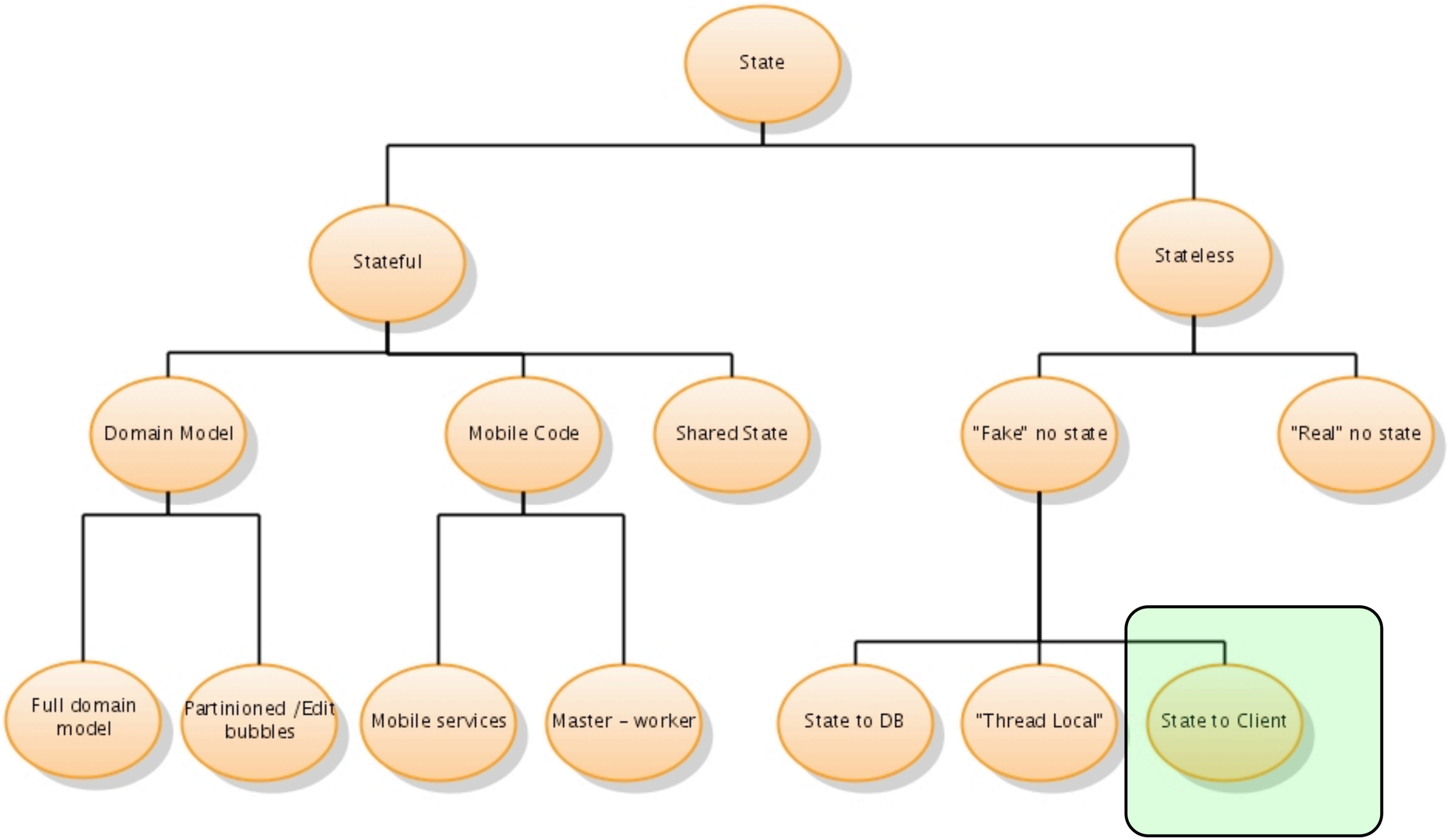
...

...

Server functionality

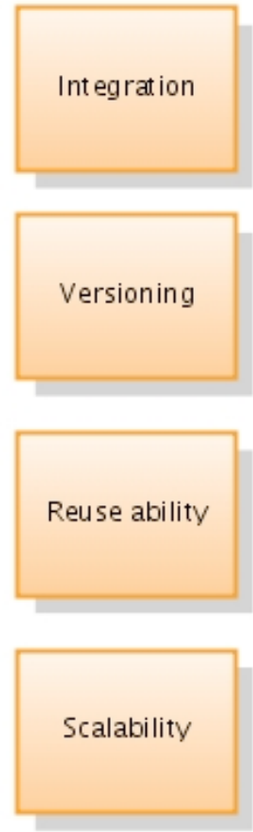
Data

Technical context breakdown

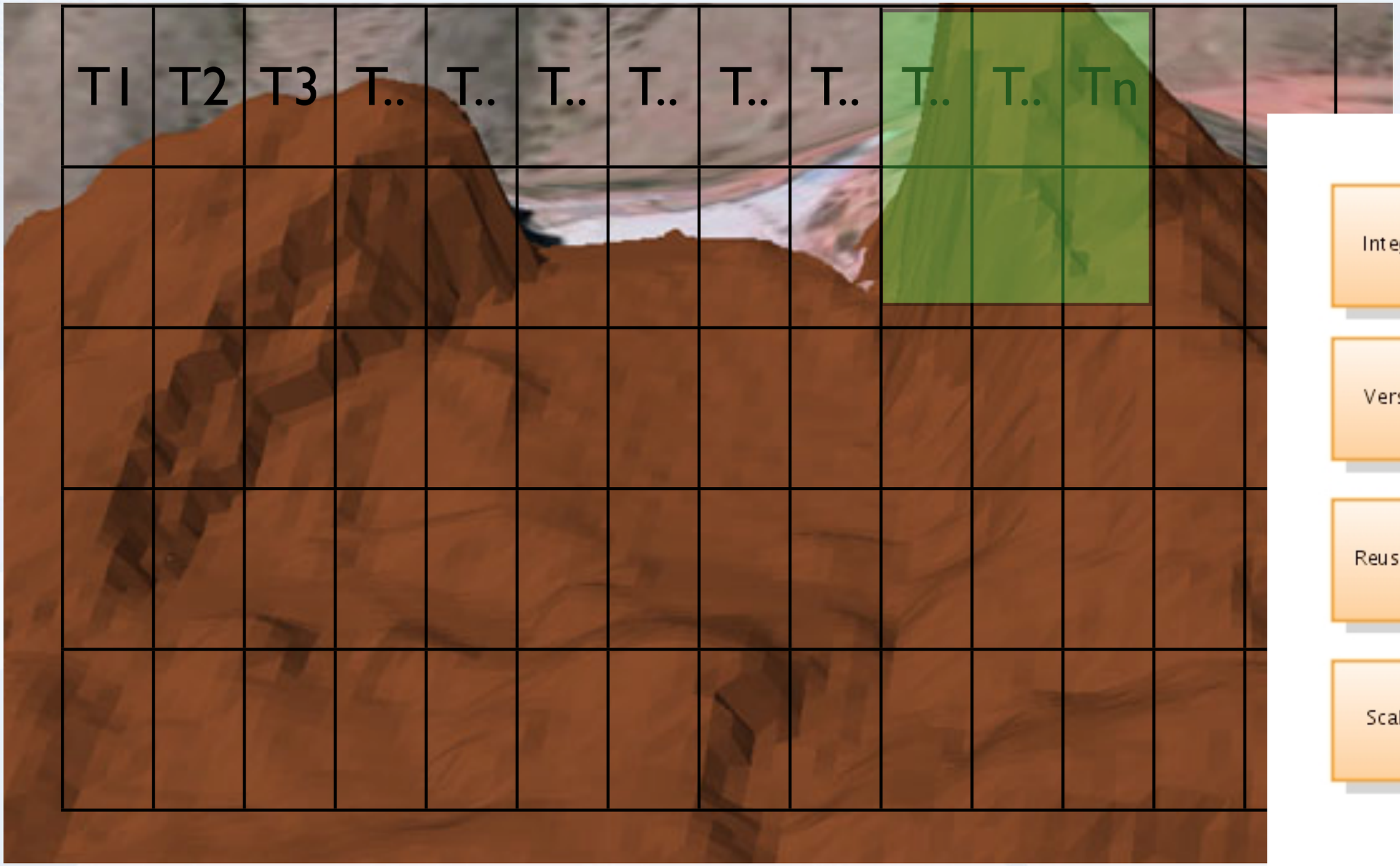


State to client

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

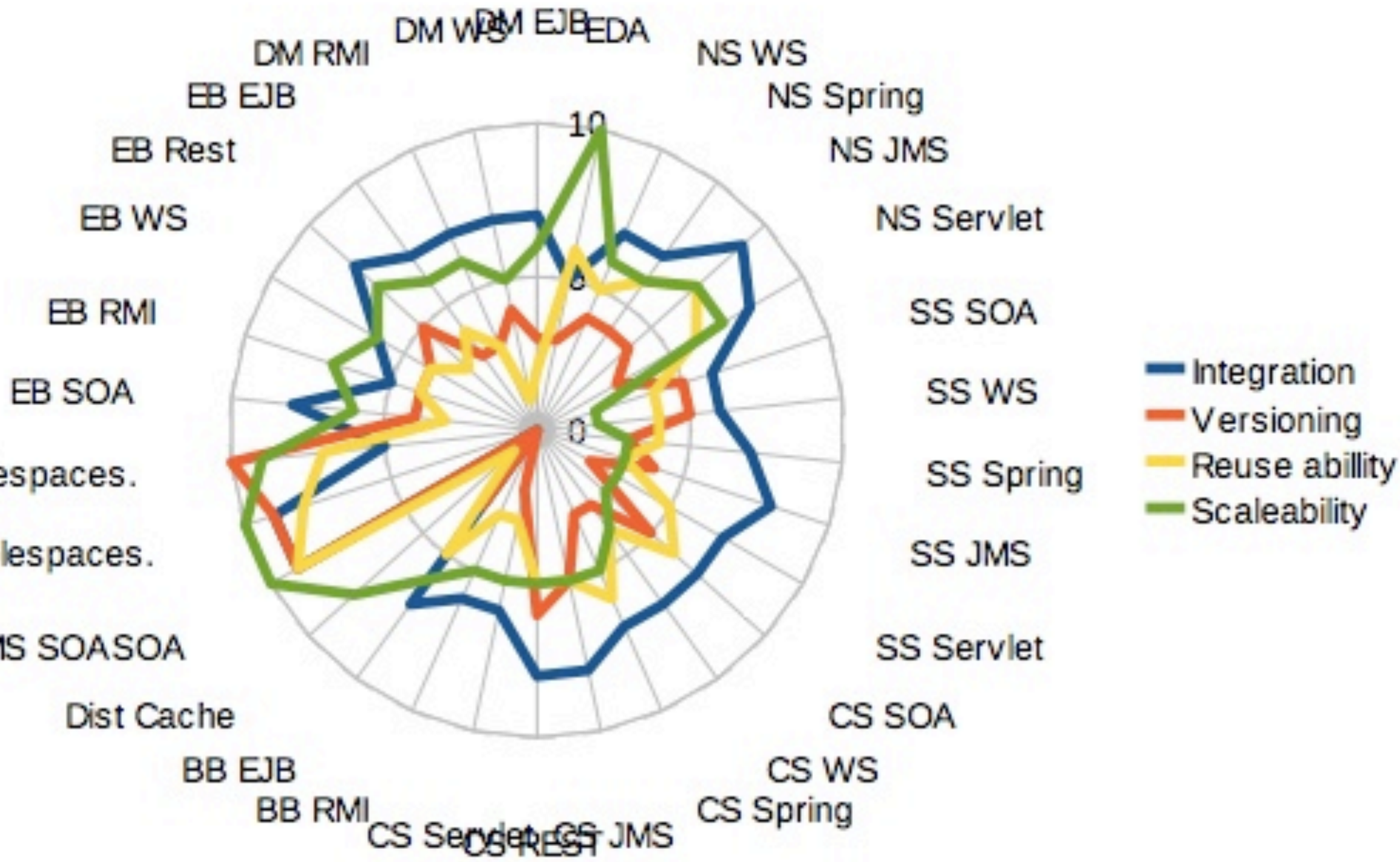


Landscapes



- Integration
- Versioning
- Reuse ability
- Scalability

Spidergraph



WTF.

- Keep focus on the right things!
- Divide and conquer
- Prioritize and rate qualities

And remember technology have different strengths in different contexts

WTF.

- Keep focus on the right things!
- Divide and conquer
- Prioritize and rate qualities

And remember technol
different strengths in c
contexts

A thought experiment

A thought experiment

- 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.
- We are free to choose **fun** and **productivity**!

Context details

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

Let´s make it fun and productive with Java and open source!

- Standard 3 layers webapp:
  - Frontend: JSF or Struts2
  - Domain logic: Spring framework
  - Persistence: Hibernate ORM to DB
- Nobody got fired from choosing IBM.

No, no, no!

No, no, no!

- **Stop and think!**

No, no, no!

- **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

JRuby on Rails

- Focus on RAD and productivity.
- Simple and productive ORM with Active Record.
- Integrates with Java.
- Built in MVC and scaffolding features.

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

Groovy on grails

- Focus on RAD and productivity
- Easy reuse of existing Java libraries
- ORM through GORM .

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

Scala and Lift

- Focus on RAD and productivity.
- Supports active record.
- Good Java integration.
- MVC support

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

4GL / model driven

- Some kind of generation based on the datamodel.
- >10 open source products available.

- Productivity factor: five to twenty times as productive.
- Fun factor: Developers hate them. below zero.

Time for a decision:

- The winner is....
- JRuby on Rails
- The best combination for fun and productivity (for this case)
- But all options are suitable alternatives.

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

Consequences

- Data mastering becomes a huge issue.
  - Not possible to do updates and/or deletes.
- 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.

Our context has changed dramatically!

Old: "Simple web-based customer system"

New: "Enterprise customer dashboard"

“Hate to tell, I told you so!”
- The Hives

- “Our Enterprise Corporate policy would have prevented this!”.
  - Wrong: No application, no problem!
  - Wrong: Poor application, same problem!

“Hate to tell, I told you so!”
- The hives

- “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.

.... and if we need to throw it away, it´s not a huge sunk cost.

Stop and think (again)

- 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,

Landscape revisited.

- 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

Landscape revisited

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

What do we do?

- Frontend: No significant changes. Re-wire to new domain objects.
- Backend:
  - Domain repository
  - Active caching of all external data.
  - Correlation of data from different sources
  - Add some new integration strategies to support the various data sources. Keep the existing.

Conclusion

- Different problems require different solutions and technologies.
- By making better selections we can:
  - Radically increase productivity and fun.
  - Improve quality of software
- Be aware of all the pitfalls
  - Find your own truth
  - Feel free to test our approach
- Make sure to have fun.

Q&A

- For more info and discussions:
- http://wiki.cantara.no
- tobiast@cantara.no
- totto@cantara.no