BEA SOA Symposium

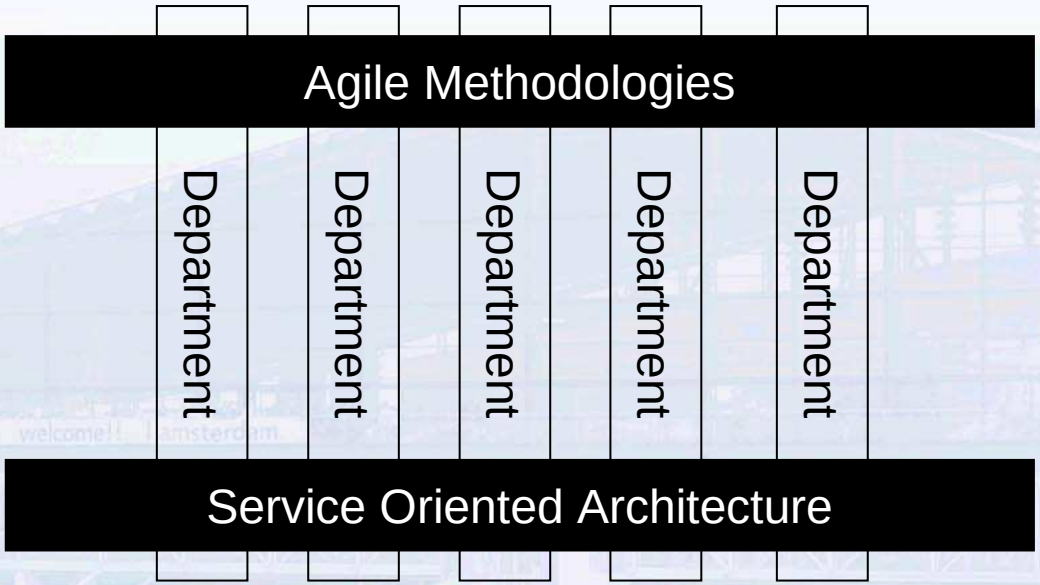
Passenger Terminal Amsterdam
Vrijdag 21 september 2007

Applying Agile principles to a Service Oriented Architecture

Vincent Partington, Serge Beaumont
Xebia IT Architects

SOA and Agile

- A SOA cuts across the organization, just like Agile methodologies do.



The Agile Principles

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Individuals and interactions over processes and tools

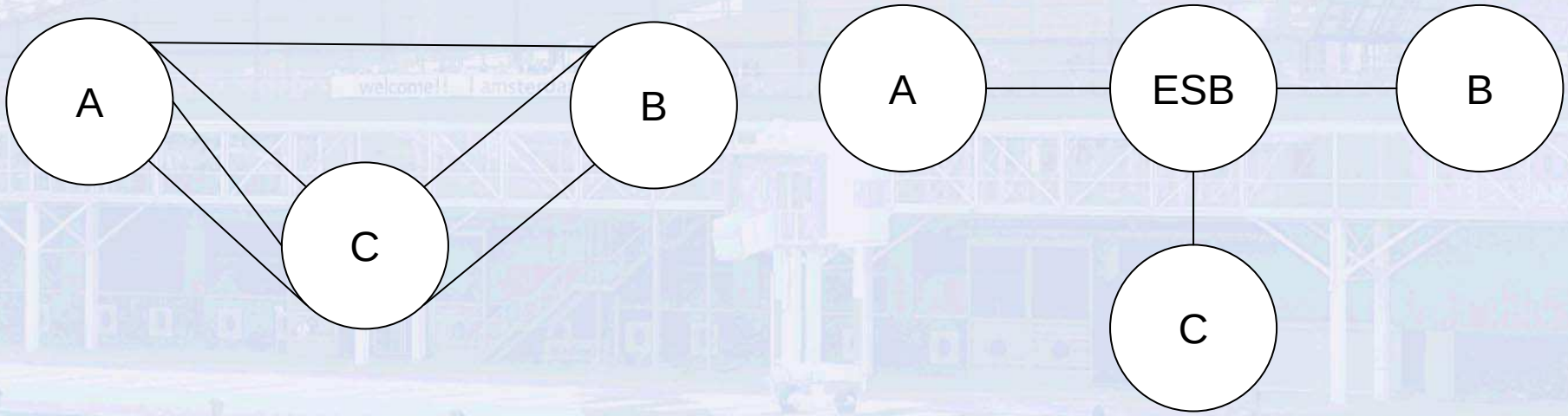
The principle

- **People create** the product, processes and tools don’t.
- Is the team a tool of the process, or is the **process a tool of the team**?
- Documents should **support** human communication, **not replace** it.

Individuals and interactions over processes and tools

The don'ts

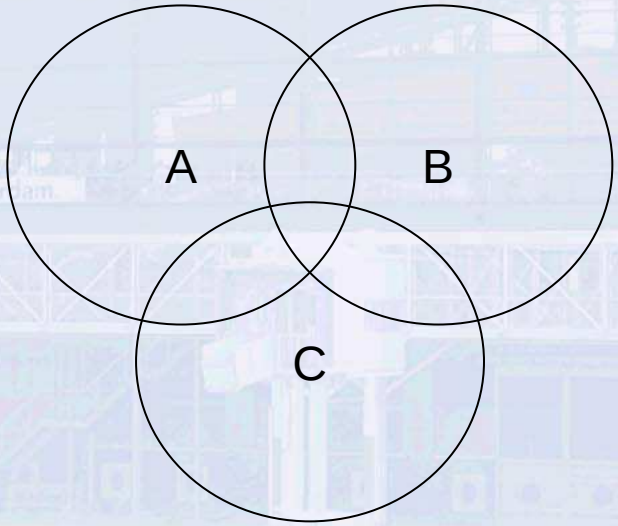
- **Don’t** start by implementing an ESB;
  - If you can’t work together already, an ESB won’t help you.
- **Don’t** build your SOA from the bottom up only;
  - Logical point-to-point services are not reusable.
  - Only the technical services (transformation, data access) will be reusable but add little to no business value.



Individuals and interactions over processes and tools

The do's

- **Do** implement an ESB for technical infrastructure;
  - Translations, Versioning, Security, Monitoring
- **Do** define a common language, i.e. a canonical model.
  - Not (necessarily) the application model!



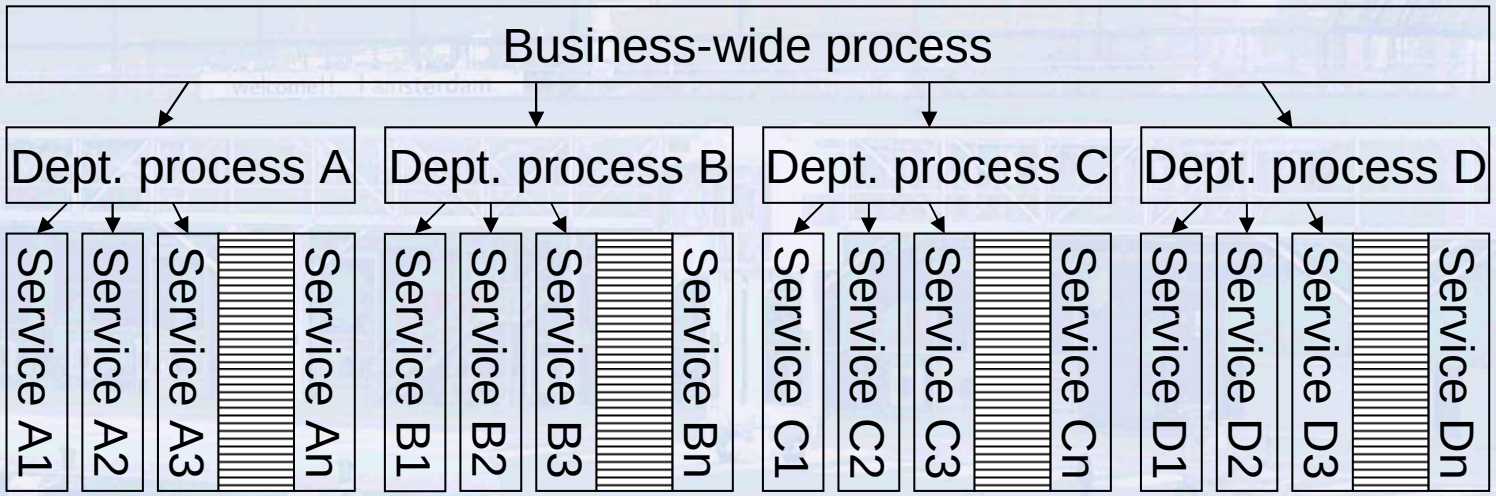
Working software over comprehensive documentation

The principle

- **Empirical** vs. predictive process
- **Feedback**: nothing beats reality
- Waste: intermediate artifacts have no **business value**
- Encounter the unexpected **early** (fail fast).

Working software over comprehensive documentation
The don'ts

- **Don’t** design the complete SOA before starting the implementation;
  - A working service is worth more than a WSDL/XSD document.
- **Don’t** design your SOA from the top down only;
  - Process-specific services are not reusable.
  - Only the data services will be reusable.



Working software over comprehensive documentation
The do's

- **Do** start small, build your SOA one service and one process at a time.
  - Risk of logical point-to-point services. Mitigate by refactoring.
- **Do** design your SOA from the top down *and* the bottom up, i.e. meet-in-the-middle.
- **Do** coordinate between the process modellers and the service designers.

Customer collaboration over contract negotiation

The principle

- Contracts are about **risk**
  - Time and Material: risk for client
  - Fixed Price: risk for supplier
  - Agile: move to **shared risk**
- From blamestorming to **brainstorming**
- Poppendieck: **measure UP**

Customer collaboration over contract negotiation

The don'ts

- **Don’t** spend all your time on service level agreements for services;
  - It’s the performance of the end-to-end process that matters.
- **Don’t** tolerate “little kingdoms”;
  - If contract negotiation takes too much time, you’ll never get a working SOA.

Customer collaboration over contract negotiation

The do’s

- **Do** create cross-departmental teams.
- **Do** measure the performance of the end-to-end process.
- **Do** have service level agreements on machines
- **Do** have service level agreements when human workflow is involved.

Responding to change over following a plan

The principle

- Waterfall: plan the work, work the plan
- **Accept reality**: we can’t predict the future
- Mass production vs. **new product development**
- Trains vs. **cars**

Responding to change over following a plan

The don'ts

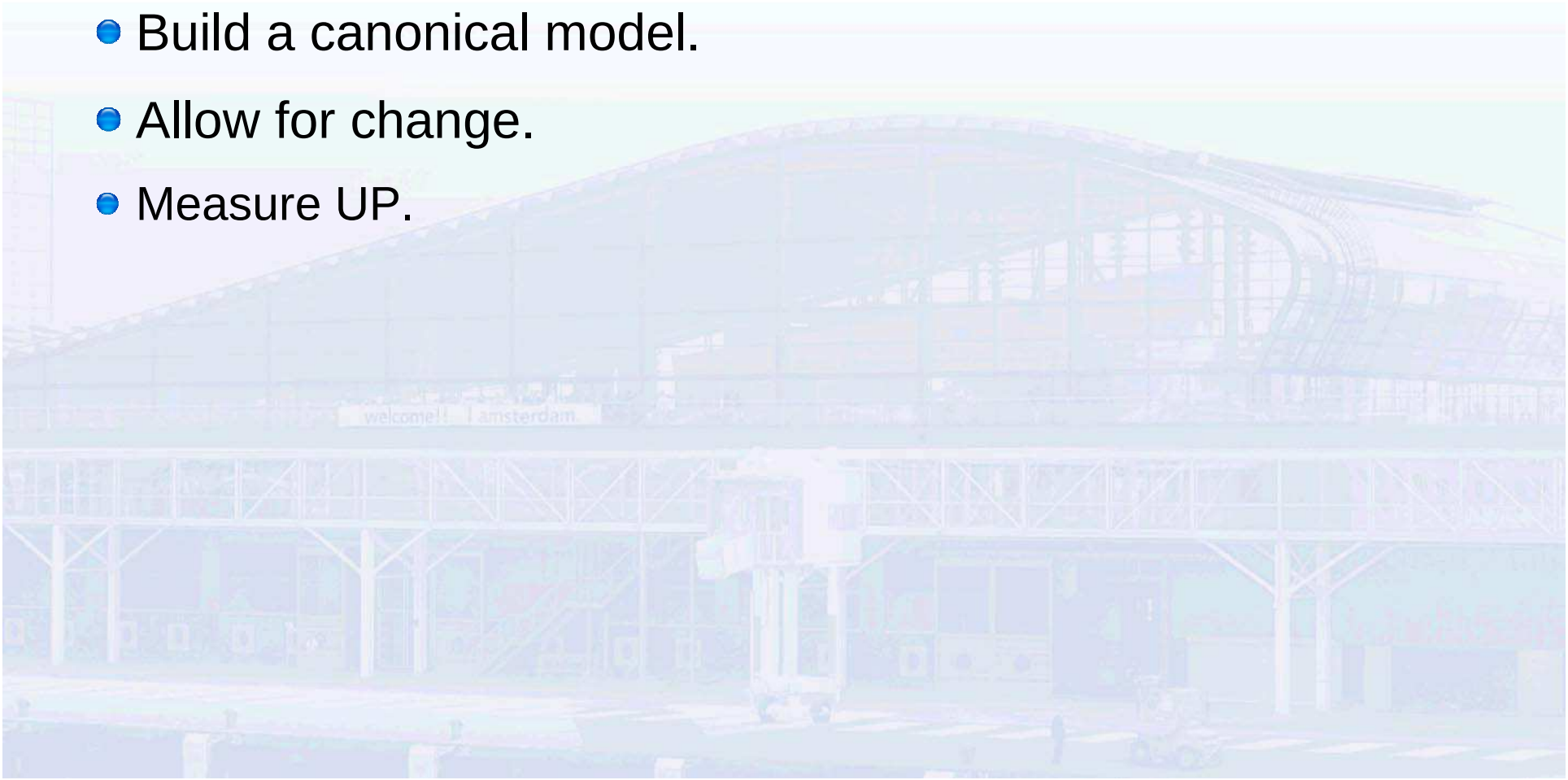
- **Don’t** assume your SOA will be finished when the project is.
- **Don’t** assume your canonical model will not change.
- **Don’t** assume you will only have one version of a service.

Responding to change over following a plan
The do's

- A SOA is about change, so **do** embrace change.
- **Do** design in lifecycle management:
  - SOA should support multiple versions of one service.
  - SOA should manage dependencies between services.
- **Do** allow the canonical model to change.
  - Canonical model is intersection of application models and will grow over time.
  - SOA should have one canonical model.
  - Translation between the canonical model and application model done by ESB.

Summary

- Start small.
- Build a canonical model.
- Allow for change.
- Measure UP.



End!

- Questions?
  - Serge Beaumont: sbeaumont@xebia.com
  - Vincent Partington: vpartington@xebia.com
- More stuff
  - http://blog.xebia.com
  - http://podcast.xebia.com