



# Improving the Extent and Reach of Service Oriented Systems

Licentiate Thesis

Martin Henkel



## Overview

- Services and Service Oriented Computing
- Identified problems – Extent and Reach
- Contribution – improving:
  - Reach and Extent – Process abstractions
  - Extent – Analysis of architectural styles
  - Reach – Use of aspects



# Software Service?

*"A well defined work that can be offered by a provider to a consumer"*

In a *Software service* the provider is represented by a software system.

Implications:

- A Software service is not used like a component
- Provider run-time responsibility



# Business and Technology Perspectives

	Concepts	Benefits with services
<i>Business perspective</i>	Business actors, Activities, Business documents, Agreements	Depict the business activities, automated business process support
<i>Technology perspective</i>	Web Services, Legacy systems, XML documents	System integration, Message routing, Message transformation, Standards -WSDL -SOAP



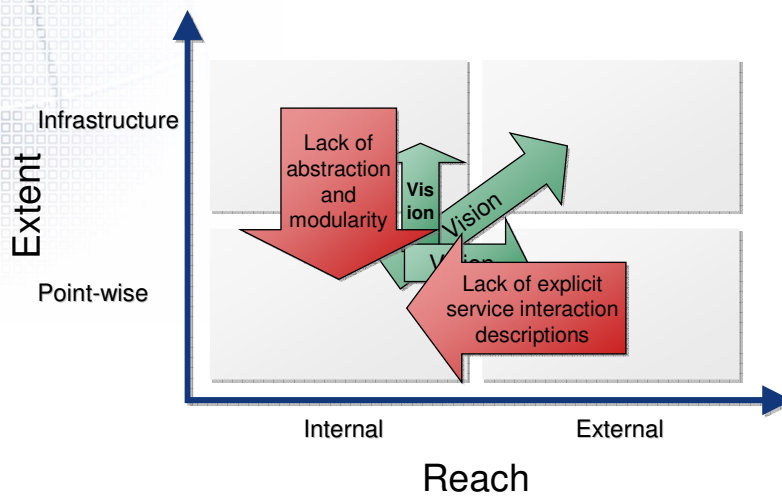
## Software Services - Vision

Use services as the fundament to support the *large scale interconnection* of existing and future systems, within a *single business* and *across business boundaries*.



5

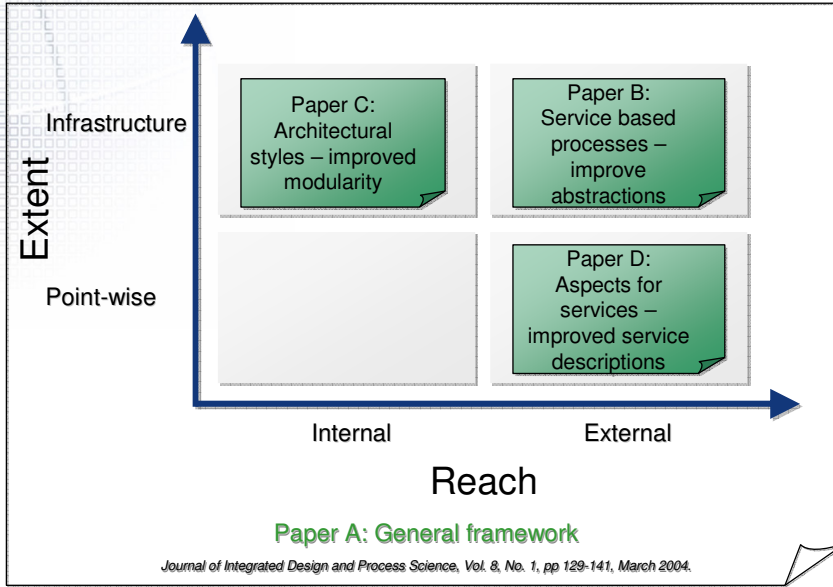
## Services – Vision and Problems



6

# Thesis Contribution

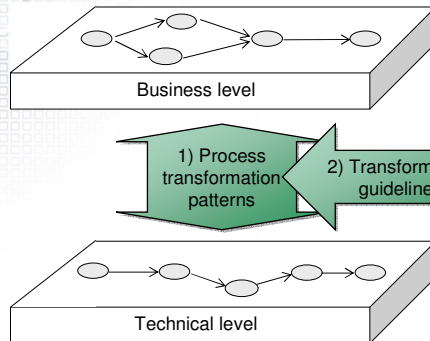
- Instruments to improve reach and extent



## Paper B Process Abstractions

*International Conference on Service Oriented Computing (ICSOC'04), ACM Press, New York, USA, November 15-18, 2004.*

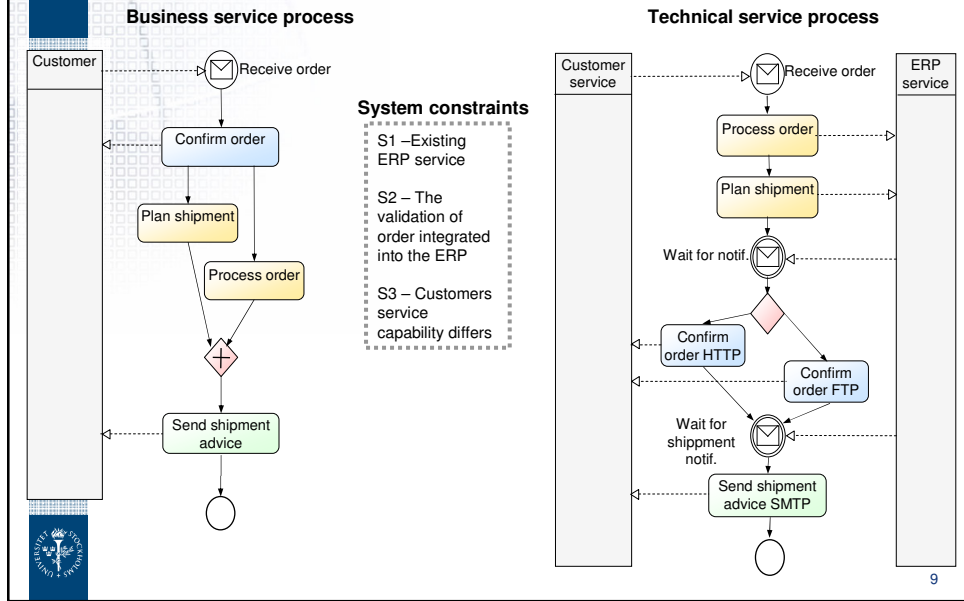
An increased extent of services could be handled if the abstraction level is raised – thereby allowing services and their interactions to be handled on a business level:



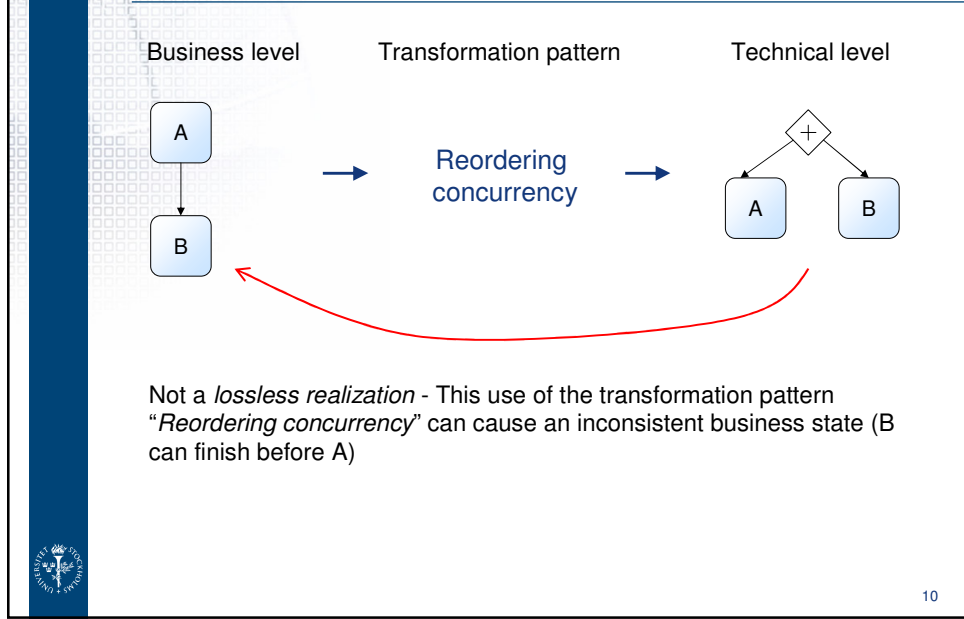
Alignment criteria: The technical implementation must be able to depict the desired states in the business layer - *Lossless* realization of the business level

Obstacle: Constraints in system technology and legacy systems makes it impossible to totally disregard the technical concerns.

# Realization – Example (based on case from Sandvik)



# Example – Concurrency Transformation



## More Rules for Lossless Realizations

Realization type	Rules for a lossless realization
Aggregation	Named activities in the technical process must be <i>aggregated</i> /mapped into a single activity name in the business process, otherwise it is not possible to determine which activity that are executing on the business level
Condition alteration (Pre & post-conditions)	<i>Conditions</i> in the technical process must be designed such that the activity pre-conditions are the same or weaker in the technical process. Post-conditions in the technical process must be the same or stronger.
Extension/Exclusion Of information	<i>Exclusion</i> of concepts in the technical process may not be introduced. However, extensions to handle technical issues might be introduced.
Condition change (branching)	<i>Conditional</i> control flow must be designed such that each (optional) path in the business process corresponds to at least one unique path in the technical process.

...



11

Paper  
C

## Architectures for Process Abstractions

The Nordic Conference on Web Services (NCWS'04), Växjö, Sweden, November 22-23, 2004.

The alignment of business and technical perspectives are a logical view of the solution.

How to implement both technical and business issues in Service oriented systems?

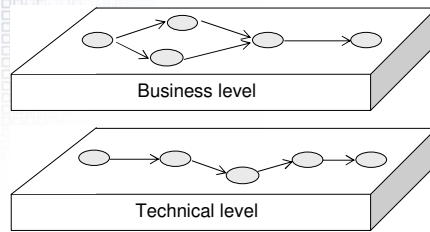
Use the transformation patterns & rules to analyse a set of typical architectural styles



12

# Layered Architecture

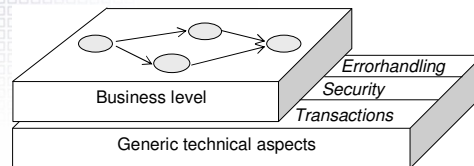
Implement the business and technical issues in two (synchronized) layers:



13

# “Aspect Oriented” Architecture

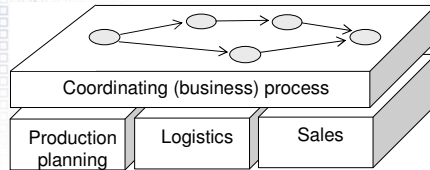
Cater to technical needs by injecting *aspects* with technical features:



14

# Domain Service Architecture

Isolate technical details into modules



15

Paper  
D

## Aspects for Services

INTEROP-ESA'05, Springer Verlag, Geneva, Switzerland, February 23 – 25, 2005.

*Aspects* can be applied to externalize a formerly internal service.  
Case example using the aspect oriented language *AspectJ*.

```
public interface CreditCheckingServiceInterface
{
    public boolean hasPaymentRemarks(String name);
    public boolean hasCreditHistory(String name);
    public boolean checkCreditForAmount(String name, int amount);
}
```



16



## Example aspect: Reliability

Aspect defined using *AspectJ*:

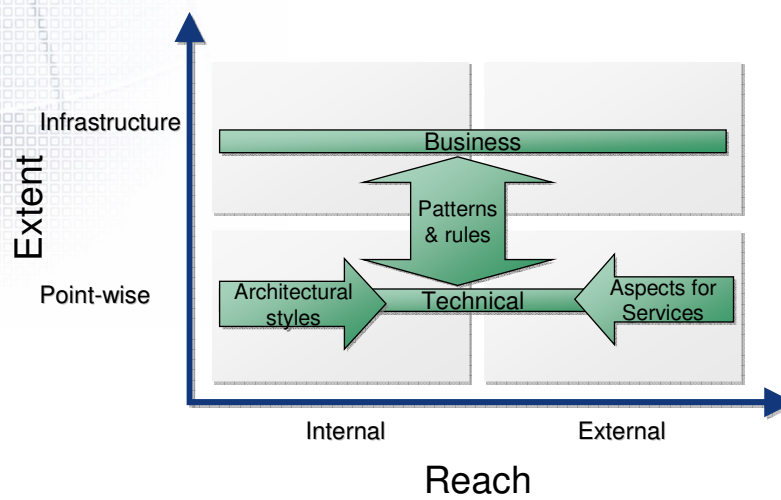
```
public aspect ReliabilityQoSAspect
{
    pointcut reliabilityMethods() : (
        execution(public * CreditCheckingService.*(..)));

    after() throwing(Exception e): reliabilityMethods()
    {
        // Log error
    }
}
```



17

## Summary of Contribution



18