Service-Oriented Software Engineering

An Introductory Lecture
Toni Ruokolainen
2006
Contents

- Introduction
- Challenges & motivation
- Service-Oriented Computing paradigm
- Characteristics of Service-Oriented Software Engineering
- Model-Driven Engineering
- Current state, trends, challenges
Introduction – Business networking perspective

Introduction - Electronic Business Networking

**eCommerce**
- Target areas: Consumer sales, procurement, distribution
- Technologies: Web Shops, portals

**eBusiness**
- Target areas: sales, procurement, distribution
- Technologies: XML, EAI platforms, Web Services, Software Components

**eCollaboration**
- Target areas: networked processes, network wide forecasting, collaborative planning, network wide product & production design
- Technologies: eBusiness standards, Integration platforms, Semantic Web, SOA

© Jyrki Haajanen, VTT
Introduction - Development of distributed computing technology

- Centralized -> Client-server -> Peer-to-peer -> Collaborative
  - From technological integration to federation
  - From tightly coupled to loosely coupled
  - From centralised to distributed control

- Collaborative systems
  - Autonomous participants, heterogeneous resources, dynamic environments
  - B2B, P2P, MAS, …

- Maturation of communication technology
  - Can now consider semantic & pragmatic interoperability
Vision: Collaborative computing

Collaboration contract
(business network model, partners, Technical details, contract breach recovery mechanisms)
Challenges

- Establishing interoperability in collaborative systems
  - Autonomy, heterogeneity, dynamism
- Bridging the semantic gap between domain concepts and technology
- Managing the complexity of computing platforms
  - Maintainability of software
    - (Re)configuration
  - Platform evolution
- Providing agility
  - Agile development methodologies
  - "On-the-fly" collaboration establishment
  - Flexibility of software systems
Motivation for research

- Support for prevailing business models
  - Outsourcing, globalisation
  - Emphasizes agility and low time-to-market
- Enhance modularity & reusability of systems
  - Provide "meaningful" abstractions for SE
  - Separation of concerns
    - Functional & non-functional aspects
- Establish more robust & effective software development
  - Model-driven engineering: productivity, agility, "fail-safety"
- Enhance maintainability of EIS
  - From monolithic systems to more flexible service-oriented systems
  - Increase ROI
- "Open service markets"
  - Fertilizes / enhances new kinds of business models?
  - SME’s role and possibilities
    - Virtual enterprises
    - Clustering
Service-Oriented Software Engineering

- Emerging software-engineering discipline
- Considers services as the primary elements of design and production
- Utilisation of Service-Oriented Computing paradigm throughout the development process
  - Emphasizes the role of meta-information
  - Development through service discovery & composition
- SOSE = SOC paradigm + SE process / methodology + SOA-based tool-chain
Service-Oriented Computing

"... (SOC) is the computing paradigm that utilizes services as fundamental elements for developing applications"

Concepts
- Service descriptions
- Service composition
- Service-Oriented Architecture

Characteristics
- Loose coupling
- Description-centricity
- Autonomy
- Collaborative computing

Service life-cycle
- Define, describe, find, bind

Copyright © 2003 by M. Papazoglou & D. Georgakopoulos
Service Descriptions

- Two kinds of descriptions
  - Service types
  - Service offers
- Service type characterises common properties for a kind of services
  - Defines syntax and semantics of a service interface
  - Ontologies, common vocabularies
- Service offer advertises a service implementation
  - Published by service provider
  - Conforms to a service type
Service Composition

- Based on service descriptions
- Results in a composite service
  - Added value (e.g. Travel company)
  - Composition synthesis
  - Composition execution
- Prevalent types of service composition
  - Process-based composition (Enterprise computing, workflow systems)
    - Orchestration, choreography
    - WS-BPEL, WS-CDL
  - Ad-hoc composition ("Roman model")
    - FSM-based automaton synthesis
  - Logic-based composition (AI, Semantic Web)
    - Pre-/post-conditions, planning
  - Architectural composition (ADL:s, CBSE)
    - Components & connectors
Service-Oriented Architecture

- An architectural style
Service-Oriented Software Engineering process

- Traditional software engineering process
  1. Analysis
  2. Design
  3. Validation and verification
  4. Implementation
  5. Maintenance

- Is this applicable for SOSE?
  - What methods / techniques are feasible in each phase?
  - What are the methods and techniques of a SOSE-process?

- Is a SOSE process different wrt. CBSE?
Service-oriented analysis

- Identify the conceptual services and collaboration patterns
  - Define service requirements
- "Services are coarse-grained objects / components"
  - Technology driven: wrapping legacy systems behind uniform service interfaces
  - Bottom-up approach
    - Providing new services
    - Identifying existing services
- "Service as a concept"
  - Ideology driven: bridging the gap between business and technology
  - Top-down approach
    - Start from business values and models
- Meet-in-the-middle
  - Impact of legacy systems
Service-oriented analysis

Example activities
- Identifying roles
- Identifying interactions
- Identifying service capabilities
  - Functionality
  - NFA
- Identifying collaborations (or service-oriented system architectures)
- Identifying pre-existing service types
Service-oriented design

- Use domain specific vocabulary to define and describe concepts given by preceding analysis phase
- Example activities
  - Creating / selecting ontologies
    - Vocabulary to be used
  - Modelling service interfaces
  - Modelling behaviour
    - Service interface behaviour
    - Business processes
  - Modelling business roles
    - Service interfaces
    - Service compositions
  - Discovery & Selection of services
    - If pre-existing services are available
Validation and verification of service-oriented software systems

- **Validation**
  - Testing service designs against design requirements
    - Simulating
    - Test generation
  - Validating conformance
    - NFA
    - Composite properties

- **Verification of designs**
  - Interoperability with prescribed collaborations and services
  - Formal verification of properties
  - Correctness of service compositions

- **Most of the standard validation procedures only applicable at runtime**
  - Need of service monitoring
  - Need of contract-driven collaboration

- **To which extent is validation of service-oriented systems possible statically?**
Implementing service-oriented systems

- Model-Driven Development
  - Both SOC and MDE emphasize the role of models as first-class entities

- Needed: A ”SOC”-programming language
  - Native XML-based messaging primitives
  - A few research prototypes exist

- Implementation technologies
  - Web Services
  - Local deployment target (J2EE, BPEL, …)
Maintenance of service-oriented software systems

- Deployment of services
- Configuration of services
- Establishment of collaborations
- Runtime management
  - Monitoring
- Evolution support
  - Platform / service evolution
  - Evolution at different levels of abstraction
Overview of Model-Driven Engineering

Domain Models → PIM → Architecture model → PIM → NFA Models

Model Transformation

Architecture style

PSM
WS specific platform models

PSM
EJB specific platform models

PSM
.NET specific platform models

PSM
other specific platform models

Model Transformation

UML profiles for: CCM, EJB, .NET, SOA...

Platform Description Model (PDM)

Model to code Transformation

Execution infrastructures

WS

EJB/J2EE

.NET/COM

Others

Copyright © 2004 by Philippe Millot (slightly edited by TR)
Hierachy of models

- MDE considers models as primary engineering artifacts
  - Models are representations of reality for a given purpose
- Models conform to a meta-model
  - Provides a typing scheme for model elements
  - Formal specification of an abstraction
- Metamodels are described with metametamodels
  - A language for defining metamodels
  - E.g. OMG’s MOF (UML is based on MOF)
MDE principles

- Three principles of MDE
  - Direct representation
  - Automation
  - Standards

Adapted from David Frankel: "An MDA Manifesto"
MDA Journal, May 2004
http://www.bptrends.com
MDE: Direct representation

- Direct representation
  - Direct coupling of problems to solutions
  - Domain-Specific (Modeling) Languages (DSML, DSL)
- DSML defines for a particular domain
  - Primary concepts and vocabulary
  - Relationships between concepts
  - Precise semantics for concepts and relationships
- E.g. specific languages for avionics, online financial services, inter-enterprise computing,…
- "Big players" in DSML field
  - UML (2.x) & UML Profiles (OMG)
  - Software Factories (Microsoft)
UML as a DSML

- Lightweight extension with profiles
  - Extends the elements of the UML meta-model
  - Profile = Set of stereotypes & Tagged values
  - "Does not provide means for precisely defining semantics associated with extensions"

- Heavyweight extension via MOF
  - Extends the UML-meta-model itself
  - All the modeling mechanisms and semantics of MOF are applicable
MDE: Automation

- Automation
  - Facets of DSL’s are intended to be automatically processed
- Transformations
  - Model-to-Model (M2M)
  - Transformation languages
    - QVT, ATL, YATL, MTL, GReAT,…
- Code generation
  - Model-to-Text (M2T)
  - E.g. PSM-to-Java, EDOC-to-EJB,…
  - QVT M2T RFP, MOFScript
MDE: Standards

- Use of open standards to establish interoperability
- Model interoperability
  - MOF, XMI
  - Meta-model bridging through unified meta-metamodel
- Infrastructure interoperability
  - Middleware infra (CORBA, WS)
  - Messaging (XML-standards, SOAP, GIOP/IIOP)
  - Communication platforms (TCP/IP)
MOF

Picture taken from http://de.wikipedia.org/wiki/Meta-Object_Facility
# Web Services

<table>
<thead>
<tr>
<th>Business Domain Specific extensions</th>
<th>Various</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed Management</td>
<td>WSDM, WS-Manageability</td>
</tr>
<tr>
<td>Provisioning</td>
<td>WS-Provisioning</td>
</tr>
<tr>
<td>Security</td>
<td>WS-Security</td>
</tr>
<tr>
<td>Security Policy</td>
<td>WS-SecurityPolicy</td>
</tr>
<tr>
<td>Secure Conversation</td>
<td>WS-SecureConversation</td>
</tr>
<tr>
<td>Trusted Message</td>
<td>WS-Trust</td>
</tr>
<tr>
<td>Federated Identity</td>
<td>WS-Federation</td>
</tr>
<tr>
<td>Portal and Presentation</td>
<td>WSRP</td>
</tr>
<tr>
<td>Asynchronous Services</td>
<td>ASAP</td>
</tr>
<tr>
<td>Transaction</td>
<td>WS-Transactions, WS-Coordination, WS-CAF</td>
</tr>
<tr>
<td>Orchestration</td>
<td>BPEL4WS, WS-CDL</td>
</tr>
<tr>
<td>Events and Notification</td>
<td>WS-Eventing, WS-Notification</td>
</tr>
<tr>
<td>Multiple message Sessions</td>
<td>WS-Enumeration, WS-Transfer</td>
</tr>
<tr>
<td>Routing/Addressing</td>
<td>WS-Addressing, WS-MessageDelivery</td>
</tr>
<tr>
<td>Reliable Messaging</td>
<td>WS-ReliableMessaging, WS-Reliability</td>
</tr>
<tr>
<td>Message Packaging</td>
<td>SOAP, MTOM</td>
</tr>
<tr>
<td>Publication and Discovery</td>
<td>UDDI, WSIL</td>
</tr>
<tr>
<td>Policy</td>
<td>WS-Policy, WS-PolicyAssertions</td>
</tr>
<tr>
<td>Base Service and Message Description</td>
<td>WSDL</td>
</tr>
<tr>
<td>Metadata Retrieval</td>
<td>WS-MetadataExchange</td>
</tr>
</tbody>
</table>

**Business Domain**
- Management
- Security
- Portal and Presentation
- Transactions and Business Process
- Messaging
- Metadata

**Copyright © 2004 by Lawrence Wilkes**
"MDE buzzwords"

MDD = Model-Driven Development
- Use of (prescribed) models and transformations during software development
- Also used commonly as a synonym for MDE

MDA = Model-Driven Architecture
- OMG initiative
- UML + MOF + QVT + few other standards..

Software factories
- Microsoft Domain-Specific Language Tools
- Software factory = model-driven product line
- Combines model-driven & component-based techniques with product-line principles
SOC & Model-Driven Engineering

- SOC provides feasible abstractions
  - “SOC complements the MDA approach by providing a template for structuring business systems and integration solutions, the service paradigm” (Piccinelli and Skene, SOSE-book, 2005)

- Both emphasize the role of meta-information
  - “The MDA complements SOC by promoting the early modeling of systems to understand their operation, a necessary prerequisite when wrapping a system with a service interface” (Piccinelli and Skene, SOSE-book, 2005)
Tools needed for SOSE

- Service-oriented analysis techniques
- Domain-specific modeling language (family)
  - Supporting viewpoints and modeling aspects
  - E.g. ODP viewpoints
- Validation & verification tools
  - Preferably weaved to design and implementation tools "transparently"
- A SOC programming language
  - Native support (type system) at least for messaging and XML-handling
- A software-engineering tool-chain natively paired with SOC infrastructure
  - Discovery of available service types / offers
  - Publication of new service types / offers
  - Deployment & configuration of service implementations
A SOA tool chain

Design
Service Modelling Tools

Implementation
Software Engineering Tools

Modelling
Enterprise Modelling Tools

Deployment
Community Configuration Facilities

Runtime
Runtime Platform

web-Pilarcos middleware interfaces

Type Repositories
uses

Service offer rep.

BNM Repositories
uses

Meta-information exchange:

Uses-relationship:
ModelBus

MODELWARE (MODELing solution for softWARE systems)
EU Information Society Technology (IST) Project
http://www.modelware-ist.org

Copyright © 2004 by Philippe Millot
Trends in research

- Concepts of SOSE
  - SOC concepts
  - Service composition
  - SOSE specific SE processes / phases
- Service & collaboration modelling languages
  - Application of formal methods
- Model-Driven Engineering
  - Model-weaving, transformation semantics
- Technology for SOC
  - Web Services
Research questions (1/4)

- Analysis & design
  - What are the concepts and methods to be used?
    - Are component-based analysis and design methods adequate?
  - What would a service-oriented analysis method look like?
  - What are the quality attributes for services?
    - Low coupling & high cohesing; How should these be interpreted in the context of service-oriented computing?
Validation & verification

- How can service-oriented designs be validated for conformance against the requirements?!
- What kind of testing methods are applicable?
  - Simulation is quite a natural choice..
- What kind of verification methods should be used?
  - Role of verification during different phases of SOSE process?
  - Verification methods before / during / after implementation
    - Design verification
    - SOC programming language type systems
    - Dynamic model checking / conformance validation
Research questions (3/4)

- Implementation
  - What kind of facilities are needed?
    - What kind of meta-information repositories?
    - Support for different composition frameworks
    - Implementation models (MDE)
    - Programming languages for SOC
  - Static / dynamic aspects of service development
    - SOC is inherently dynamic environment
    - What can be done before / during maintenance phase?
    - How to identify this distinction during the engineering process?
    - How to address dynamic aspects?
      - Are they modeled explicitly? Left implicit and dealt in the platform level?
  - MDE challenges
    - Viewpoint modelling & correspondences / relationships between views
    - Aspect languages & model weaving (esp. NFA)
    - Static & dynamic use of meta-information
Research questions (4/4)

n Maintenance
  n Deployment of service-oriented systems
    n Deployment of complex systems is painful
      \( \text{尤其是如果配置需要经常更改} \)
      n What kind of configuration mechanisms are needed?
  n Establishment and maintenance of service-oriented collaborations
    n Facilities for establishing and upholding contracts
  n Evolution management
    n What kind of changes can be tolerated?
    n How? At which points of time?
    n At which level? (implementation, model, meta-model)
Conclusion

- SOSE is still an open field
  - Results stemming primarily from OOSE & CBSE
  - There is no complete SOSE process available, yet?
- SOC has been quite technology driven
  - Strong role of industry in Web Services standardisation
  - The role of SOC infrastructure in the software engineering process has been left quite vague
- MDE is a good fit for SOSE
  - Both emphasize the role of meta-information
  - Quite easy to generate (at least simple) skeleton code for service-interfaces
  - In infancy: model (aspect) weaving, compositionality, separation of concerns (viewpoints)
- SOC & SOSE emphasize convergence between "business" & infrastructure
  - There is no strict phasing of a SOSE process but inter-dependencies instead?
Practicalities

- Contact: Toni.Ruokolainen@cs.Helsinki.FI
  - Room: D219

- Timetable for the next few weeks
  - 12.09.2006 No session
  - 19.09.2006 Abstracts due!
    - PDF-format, 1-2 pages (look at the seminar page for more information)
  - 26.09.2006 No session
  - 03.10.2006 First presentation

- Reservation of time-slots
  - Now..
  - Opponents(?)