Software for WorkFlow Management

Workflow Management Systems

Virtual Enterprises:
Web Services, SOA, WS-BPEL etc
Recap on Workflow

- Workflow (definition from WorkFlow Management Coalition):
  - “The computerised facilitation/automation of a BP, in whole or part”
  - Workflow technology is often an appropriate solution to BPR activities.
  - Traditionally managed by software (Workflow Management Systems WFMS)

- Thus workflows involve the coordinated execution of multiple ‘tasks’/‘activities’ performed by different processing entities, nowadays mostly in distributed heterogeneous environments

- These are very common in enterprises of even moderate complexity

- A workflow system can be defined as a collection of processing steps organized to accomplish some BP
Recap on Workflow (cont’d)

• Note: A task may represent
  – a manual operation by a human or
  – a computerizible task to (a) execute legacy applications, (b) access databases, (c) control instrumentation, (d) sense events in the external world, or (e) even affect physical changes

• In addition to the collection of tasks, a workflow defines the order of task invocation or condition(s) under which tasks must be invoked (i.e. control-flow) and data-flow between these tasks

• Workflow is the process by which individual tasks come together to complete a transaction - a clearly defined business process - within an enterprise.
Recap on Workflow: (cont’d) - WFMS

Workflow System Characteristics

[Diagram showing the workflow system characteristics with roles and processes such as Process Designer, Administrator, Business Process Analysis, Process Design & Definition, Build Time, Run Time, Process Instanciation & Control, Workflow Enactment Service, Applications & IT Tools, Interaction with Users & Application Tools, and process changes.]

CA441: WFMS, SOAs & Web Services
Changes in Context: (cont’d)
Recent Broad Goals/Trends

Goals:
- Low cost
- Streamlined & efficient process
- Monitor & track process execution
- Detect and manage exception
- In-time response, etc
- **Solution**: IT

Business Trends
- **Scalewise**:
  - Intra-Enterprise
  - Inter-Enterprise
  - Global Interaction

IT Trends
- Mainframe
- Set of Servers
- Set of Services

CA441: WFMS. SOAs & Web Services
Changes in Context: (cont’d)
Problems & their Current Solution

• Different parties (even in the same company) may have different
  – Operating system, interface, data format, infrastructure, interaction protocols, language, etc

• Automating *supply chain* implies bringing all of these together

• The Solution to this problem is Integration

• Current Integration solution is *Enterprise Application Integration*

*Comprised of interaction btw parties required to produce products/services & deliver them to customers*
Changes in Context: (cont’d) Terminology

• **Enterprise Architecture (EA)**
  – Informally: A picture of the enterprise in terms of the BPs, data model, organisational structure etc...

• **Enterprise Application Integration (EAI) technology**: the means of integrating existing s/w systems or applications within enterprises with each other in order to execute BPs involving many s/w systems
  – User Interface Integration,
  – Data Integration
  – Method or Function Integration
  – Business Process Integration

• **Middleware** is the communication facilitator in EAI. The *Enterprise Service Bus (ESB)* is a common realization of this (more later).
EAI: (Cont’d) Three Views

The Low Level View

- Client application
- Server-side application

Communication middleware

- Server-side application

Enterprise Architecture

- EAI: Enterprise Service Bus

Services

EAI technology

- Business process step
- Data transformation

Adapte

CA441: WFMS, SOAs & Web Services
More Changes in Context:
Specific Business Challenges to BPM

• Methods of Business Process Management are useful when optimising BPs within an enterprise.

• Problems come when trying to extend an enterprise: BPM is neither scalable nor adaptive by nature
  – So a framework based only on BPM can be used to build business applications but they will be so cohesive as to be inflexible to adapt to future changes.
  – For each change, business dept must interact with IT dept to develop software changes.

• As explained, SOA supports agility in software development through a loose coupling of services thus obviating the need to talk to IT

• Still need BPM as processes will need to be optimised (Bajwa et al 2008)
More Changes in Context:
Specific Business Challenges to BPM (cont’d)

• Need for increased agility in business processes and collaboration in loosely-coupled networks (*Virtual enterprises*) (Kutvonen, 2005)

• Some business environments require many different process designs (Smith and Fingar, 2004)

• Mass-customization of processes => Automation of process creation (Example: patient health records)

• Processes, which evolve dynamically as they execute, through the exchange of information among participants whose relationships evolve as a result (Smith and Fingar, 2004)
Changes in Context: The Virtual Enterprise

- Virtual Enterprise (VE): temporary alliance of businesses coming together to share skills or core competencies & resources to better respond to business opportunities, and whose cooperation is supported by computer networks.
- Generally SMEs but can include big companies (e.g. Nike or IBM)
- Usually they “buy” services and things rather than “build” or “make” them
Typical *status quo* in Many Enterprise IT Architectures

- Functional and technical **application monoliths** ubiquitous
  - Stovepipe* architectures, application scope creep, redundant implementations, data management and many other agility issues
  - Architectural governance or guidance missing

- Development and integration projects **costly and long running**
  - Proprietary point-to-point connections, often developed from scratch
  - File transfer is a frequently used integration pattern with numerous architectural drawbacks
  - ‘Roll-your-own’ philosophy works short term, but leads to maintenance headaches

- As a result, horizontal initiatives are **much harder to implement** than they have to be
  - Example: single customer relationship management solution on top of several line-of-business applications (packages and custom developed)

---

*refers to "islands of automation" in an enterprise, designed independently with little commonality/interoperability.*
Evolution into a Service-Oriented Architecture (SOA) Ecosystem

Component-Based Development

Messaging Backbone

- Point-to-Point connection between applications
- Simple, basic connectivity

Enterprise Application Integration (EAI)

- EAI connects applications via a centralized hub
- Easier to manage larger number of connections

Service-Oriented Architecture

- Integration and choreography of services through an Enterprise Service Bus (ESB)
- Flexible connections with well defined, standards-based interfaces

Source: IBM SOA

© IBM Corporation
But what is SOA, Anyhow?

- Its an architectural style whose goal is to achieve “loose coupling” among interacting & contracted services via communication protocol.

- Often seen as built on, & evolving from older concepts of distributed computing/ modular programming.

- Formally “refers to an architectural style of building reliable distributed systems that deliver functionality as services, with the additional emphasis on loose coupling between interacting services.” OGSA Glossary.
But what is SOA, Anyhow? (cont’d)

• Architecture is not tied to a specific technology

• SOA is commonly built using Web services* standards

• Can also be implemented using any service-based technology at a higher cost

• The model and the notation followed in this architecture mimics what has been done in traditional RPC technologies

• First implementations are just extensions of existing platforms to accept invocations through web service interfaces

*more later
SOA fundamentals: Modularity, layering, and loose coupling

Example:
An insurance company uses three SAP R/3, MS Visual Basic, and COBOL applications to manage customer information, check for fraud, and calculate payments. The user interfaces (UIs) are the only access points.

A multi-step, multi-user business process for claim handling, executing in IBM WebSphere, is supposed to reuse the functions in the existing applications. How to integrate the new business process with the three legacy applications in a flexible, secure, and reliable way?
SOA Principle 1: Modularity (i.e. Separation of Concerns)

• Motivation:
  – Integrating monolithic applications (“stovepipes”) is hard (e.g., traditional Enterprise Resource Planning packages)

• Solution
  – Refactor to services, expose service interface only, hide implementation details (a.k.a. encapsulation)

• Consequences
  – Service contracts have to be defined and interpreted (by tools and/or at runtime)
  – Services have to be located and invoked in a coordinated manner
  – Service invocations have to be free of undesired side effects (data mgmt?)
SOA principle 2: Layering (logical and/or physical)

• Motivation
  – Service characteristics such as interface granularity & lifecycle vary: e.g. technical logging service vs. claim checking business process

• Solution
  – Organize SOA into 3++ architectural layers

• Consequences
  – More abstraction (i.e. services can be composed out of other services leading to composite applications), requiring communications infrastructure
  – First law of distribution: “the best remote call is the one you don’t make”
SOA principle 3: Loose coupling through messaging

• Motivation
  – Once applications have been modularized, dependencies between services occur

• Solution
  – Couple services loosely (several dimensions: location, time, invocation context)
  – E.g. messaging system decouples in time, location, language dimensions

• Consequences
  – Messaging means single impln/endpoint by default (no remote objects)
  – Asynchronous communication complicates systems management
SOA principle n: Service virtualization and flexible infrastructure

• Motivation
  – “I don’t care about a particular provider, just chose the one that at this point in time is best for me”

• Solution
  – From WWW to service bus/cloud
  – Two-level programming

• Consequences
  – Many open issues e.g., trust and privacy, precise semantics, QoS, multi tenancy

• First isolated steps
  – Software as a service (SAAS), e.g. Salesforce.com CRM & Amazon Storage Service
  – Dynamic matchmaking, grid and utility computing, on demand
Slide Aside: What is *virtualization*?

- A simple example of virtualization: my desktop

  - My old (physical) machine (XP, applications & all) has been virtualized using VMWare software - This is *hardware* virtualization (see fig)
  - Another form of virtualization you will be familiar with is *software* virtualization (e.g. Java Virtual Machine or JVM)
  - There are many other forms of virtualization.
Recall: Changes in Context: (cont’d)
Recent Broad Business Goals/Trends

Goals:

- Low cost
- Streamlined & efficient process
- Monitor & track process execution
- Detect and manage exception
- In-time response, etc

Solution: IT

IT Trends

- Mainframe
- Set of Servers
- Set of Services

Business Trends

- Scalewise:
  - Intra-Enterprise
  - Inter-Enterprise
  - Global Interaction

- Timewise
  - Manual
  - Electronic
  - Web

CA441: WFMS. SOAs & Web Services
SOA in Practice: Use of Web Services

• Web Services = SOA + Standards (WSDL/SOAP/UDDI/XML etc)

• Web Services is the natural evolution of middleware & EAI platforms as they try to leverage:
  – the Web
  – the Internet
  – the globalization of society, particularly in its economic aspects

• No difference from middleware except:
  – being invoked via Internet

• A **standardized** means of dealing with integration, where traditional methods are vendors/application/language specific
What are Web Services, Anyhow?

• Web Services is a model for using the Web:
  – To automatically initiate processes via the Web using programs
  – A method for describing, publishing & initiating processes dynamically in a distributed environment
  – Not necessarily using a Web browser
  – Actually, the Web is not required?

• **Content-oriented Web** now complimented by **Service-oriented Web**

• If you can imagine a way of electronically delivering something:
  – Of value to a customer
  – That will solve a problem, or
  – Provide some service to them

Then you have a viable example of a Web service!
More on Web Services

• To put it simply, a Web service is nothing but a server that listens for and replies with SOAP generally via HTTP

• A Web Service is an interface that describes:
  – a collection of operations
  – that are network accessible
  – thro standardized XML Messaging

• Web services encapsulate business functions:
  – Check credit card number, Payment processing, Stock quotes
  – Request for quote, bid processes

• They can be used to compose business processes
  – Travel planning, Health care, Etc, etc
More on Web Services

- Benefits of Web Services Include:
  - Decoupling of service interfaces from the implementation
  - Enabling dynamic service binding
  - Providing interoperability among different platforms
  - Existing applications can be wrapped as Web services
  - Client & Service can use different platforms & programming languages
  - Services can be composed to make composite services
“Publish-Find-Bind” Model of Web Services

Discovery Agency

Service Specification

Query

Publish

Service Requestor

Interact

Request

Response

Service Provider

Service Specification

Service

Requirements

Request

Response
Web Services: Technical Bits

- WS technologies use eXtensible Markup Language (XML) documents to describe, invoke, and publish services.

![Diagram of Web Services architecture]

**WSDL**: Web Services Description Language

**SOAP**: (formerly) Simple Object Access Protocol

**UDDI**: Universal Description, Discovery, & Integration

---

**Module 1**

- Contract Repository
- Service Broker
- Service Provider
- Web Service
- Client Application

**Module 2**

- Service Requester
- UDDI Registry

**Flow:**

- Contract Repository → Service Broker → Service Provider → Web Service → Client Application
- Service Requester → UDDI Registry → find WSDL → Service Provider → bind, invoke

---

Source: [IBM ITSO]

- WS technologies use eXtensible Markup Language (XML) documents to describe, invoke, and publish services.

---

**UDDI**

- Publish WSDL
- Find WSDL
- Bind, invoke via XML messaging

**WS:**

- WSDL: Web Services Description Language
- SOAP: (formerly) Simple Object Access Protocol
- UDDI: Universal Description, Discovery, & Integration
Web Services: Technical Bits

<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
      <n:priority>1</n:priority>
      <n:expires>2001-06-22T14:00:00Z</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Pick up John at school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>

• An example of an XML-based SOAP message. Requestor, Provider communicate using messages such as these.
But what has all these Acronyms to do with Workflow & Business Processes?

- **WS-Policy**: a specification allowing WS to advertise their policies (on security, QoS etc.) & for WS consumers to specify their policy requirements.
- **WS-BPEL** or BPEL4WS: a language for definition & execution of BPs using WS.
Business Process Execution Language (BPEL)

- **WS-BPEL**: a language for describing BPs based on Web Services
  - Processes described using WS-BPEL execute functionality by using Web Service interfaces exclusively
  - WS-BPEL Specification is administered by OASIS

- **WS-BPEL** is an *orchestration* language, not a *choreography* language
  - *Orchestration* specifies an executable process that involves message exchanges with other systems, such that the message exchange sequences are controlled by the orchestration designer.
  - *Choreography* specifies a protocol for peer-to-peer interactions, defining the legal sequences of messages exchanged with the objective of guaranteeing interoperability.
  - Orchestration means that it actively describes the ways in which individual services can be composed to implement a more complex service.

- **WS- BPEL** can integrate external services as well as human interactions, so that typical business processes can be easily mapped to BPEL descriptions.

- In most cases, users have a tool for designing and validating business processes, and another one for executing these processes.
BPEL (Cont’d)

• BPEL processes are executed rather than observed
  – they can be started by external actions
  – they can be started by some internal process (within the BPEL context)

• BPEL processes can be long-running transactions
  – if human interaction is required, processes may take weeks or months
  – BPEL engines store process state persistently to preserve the state information

• BPEL processes are described like flow charts
  – BPEL defines a small number of basic building blocks
  – special controls are used for branching, joining, and exceptions
  – BPEL execution is the process of transporting data through the chart
Example of WS-BPEL Code: Hello World!

```xml
<?xml version="1.0" encoding="UTF-8"?>
<process
   xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/
   xmlns:print="http://www.eclipse.org/tptp/choreography/2004/engine/Print">
   <!--Hello World - my first ever BPEL program -->
   <import importType="http://schemas.xmlsoap.org/wsd/"
      location="../../test_bucket/service_libraries/tptp_EnginePrinterPort.wsdl"
      namespace="http://www.eclipse.org/tptp/choreography/2004/engine/Print" />

   <partnerLinks> <!--What the WS is called and where it is found-->
      <partnerLink name="printService"
         partnerLinkType="print:printLink"
         partnerRole="printService"/>
   </partnerLinks>

   <variables> <!--What is the message variable called and what is its type-->
      <variable name="hello_world"
         messageType="print:PrintMessage"/>
   </variables>

   <assign> <!--What is the content of the message -->
      <copy>
         <from><literal>Hello World</literal></from>
         <to>$hello_world.value</to>
      </copy>
   </assign>

   <invoke partnerLink="printService" operation="print" inputVariable="hello_world"/>
</process>
```
Example 1: A Simple WS for the Travel Booking Process
Example 2: WS & SOA in the Banking Industry

• Brahe (2007) shows in a case study how SOA was adopted in Danske Bank.

• Danske targeted all application development on SOA & transformed legacy systems into web services by wrapping them up.

• With a central service library & repository, applications & services developed for one part of the group can be located and reused by each other part of the group.

• BPs were designed using an IBM BPM system & implemented using BPEL.
Example 2: WS & SOA in the Banking Industry

• The diagram shows a sample BP consisting of different actions.
• Each action is either an automated task, implemented by a WS or a human task, performed by a user through a user interface of the system.
• The figure shows how different types of applications (COBOL, Java, SAP etc.) can be wrapped into WSs & therefore integrated into consistent processes and combined with human tasks.
**Example 2: WS & SOA in the Banking Industry**

- Danske Bank tried to produce *customer packages* as a pilot to see how combining SOA/BPM influences their business performance.

- *Customer packages* were a new sales concept, where customers can sign up for packages containing e.g. an account, a credit card and an internet bank account.

- The package is then created by back office people from different legacy systems in a highly predictable and production-like way.

- Since the process involved applications from different departments, a WF would be able to link different applications together to integrate them in a continuous end-to-end process.

- Following BPM methods, Danske tried to model the existing BP in a WF....
Example 2: Problems & Resolutions

• However they soon found the process to be more complicated than they thought.

• As they had never tried to describe the BP in detail before, the first WF version only contained the *main* path thro the BP, without exceptions and special cases.

• In the end, the model contained ~30 different WFs & 200 service invocations or human tasks.

• After orchestrating the different task into an integrated WF system, Danske started to automate some tasks by developing automatic product creation services.

• They did not choose a radical BPR approach, as they knew that stepwise optimisation allows for the chance to learn from experience & execution statistics.

• Also, since they could implement changes in a controlled matter, this gave the back office workers more time to adopt to changes.
Example 2: Conclusions

• Danske was able to achieve the following efficiencies:
  – Reducing the rate of manually created products to no more than 20%.
  – Automating data-flow between different systems, thus making back office employees’ work easier and more efficient.

• Besides those benefits, Brahe also discovered various challenges, which arise with the application of BPM and SOA:
  – It’s crucial for easy integration of different services that these have been designed for reusability and (especially) the documentation has been done properly.
  – BPM and SOA are concepts, methods & techniques that are not easy to adopt, i.e. there has to be a commitment not only to technical but also educational effort.
  – Firms have to study best practice examples and need to keep a strong architectural governance to ensure that all future projects follow the given development process and standards as well as the service-oriented guidelines.
  – Existing commercial standards and tools were not yet mature enough to support a model-driven and service-oriented development process efficiently.
References


