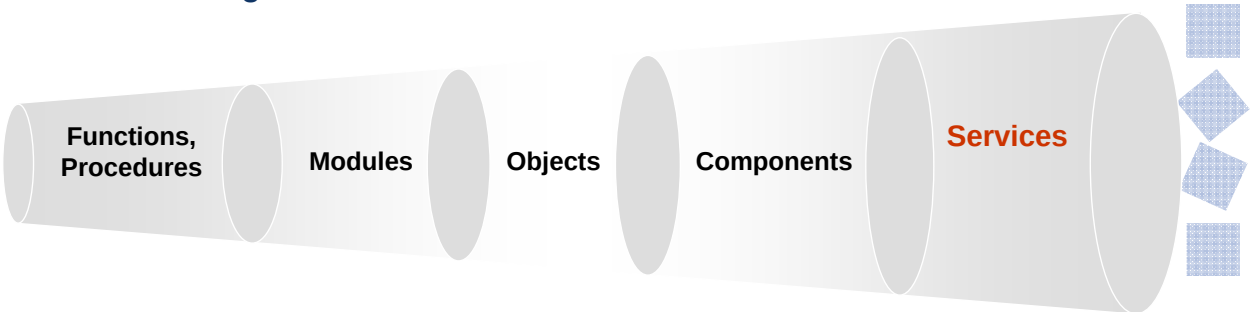
Service-Oriented Architecture (SOA)

Marlon Dumas

27 November 2007

Context: Bridging IT & Business Value

- Closer communication between business & IT
  → sharing common concepts and models
- Shift from “programming in the small” to “programming in the large” with emphasis on:
  - Integration
  - Adaptation
  - Configuration



What is SOA? – Purpose

- *SOA is a paradigm for organizing and utilizing distributed resources and capabilities that may be under the control of different ownership domains.*
  - » *OASIS SOA Reference Model*
- SOA does not imply “technology”, but it can help to structure how technology is deployed and organised.

What is SOA? – From a Software Engineering Perspective

SOA is a structure for a system consisting of a collection of software services.

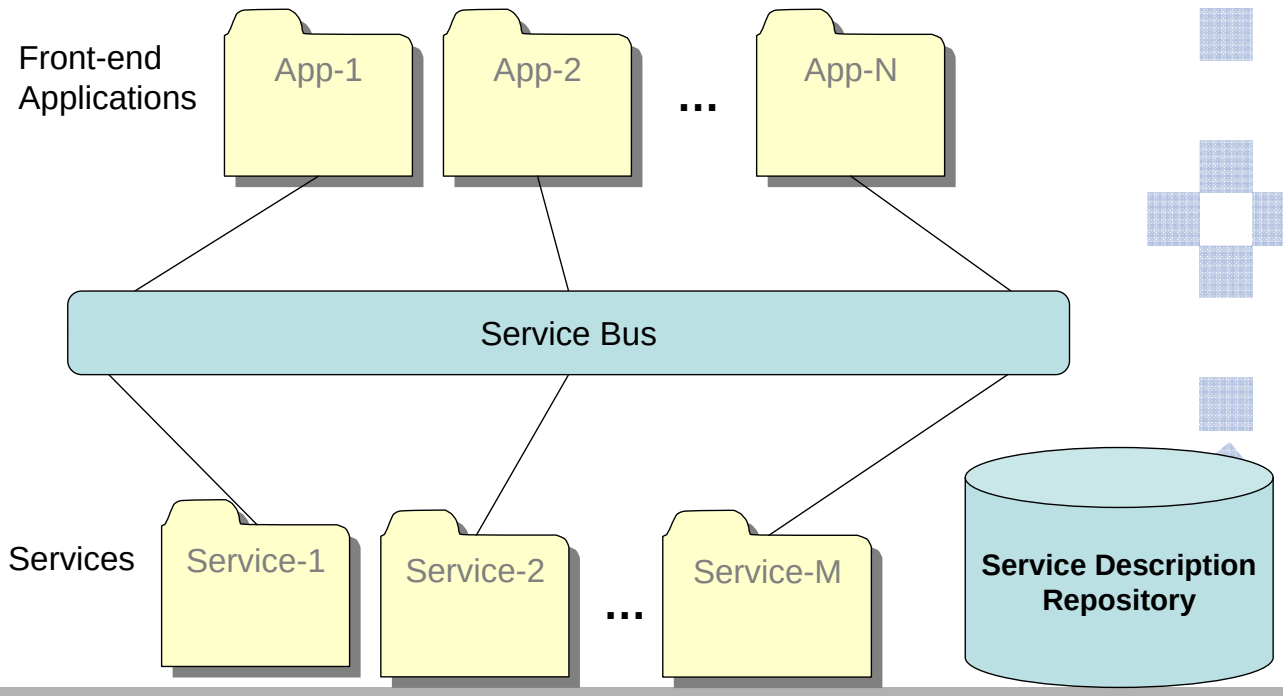
So what’s a *software service*?

- A software asset that is deployed at an endpoint and is continuously maintained by a provider for use by one or multiple clients.
- Services have explicit contracts that establish their purpose and how they should be used.
- Software services are (supposed to be) reusable.

SOA Principles

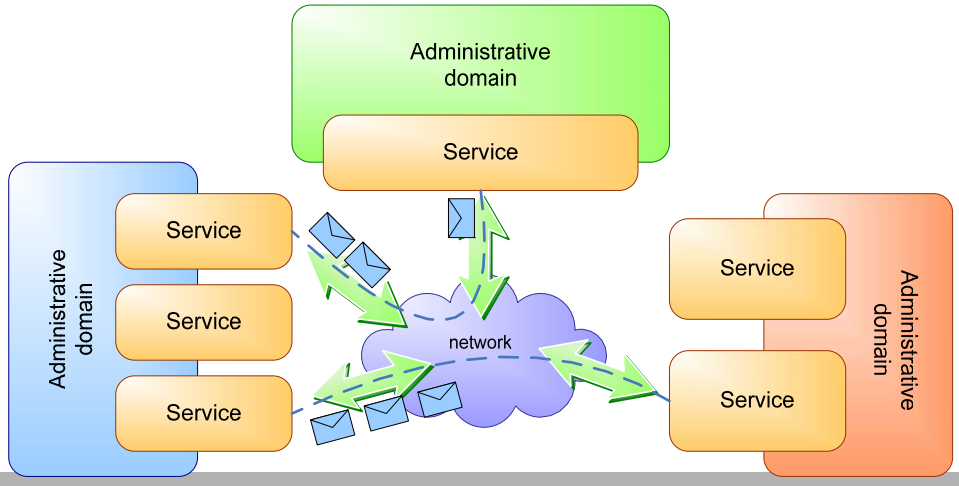
- Abstraction
  - Services share standards and contracts and nothing else!
- Autonomy (loose-coupling)
  - Minimize dependencies between services and make dependencies explicit – Services have owners!
- Reusability
  - Services are intended to be reused and composed.

Components of a SOA (Simplistic View)



The Service Bus

- Connects services, front-ends and repositories in an SOA
- It is not a software product (even though some vendors would like to sell it as such!):
  - Needs to support a variety of technologies & communication modes
  - Needs to be cross administrative domains…



Types of Service

- Application front-end
- Basic service
- Intermediary service
- Process centric service
- Public enterprise service

Although they are not services themselves – are the active Elements of a SOA. They initiate all business processes and ultimately receive their results.

Are the foundation of the SOA. They represent the basic elements of the vertical domain. Include both data centric and logic centric services.

Include technology gateways, adapters, facades and functionality-adding services. Are both client and server in a SOA. Unlike process-centric services they are stateless.

Encapsulate the knowledge of the organization's processes. Typically both client and server in a SOA, and maintain the process state.

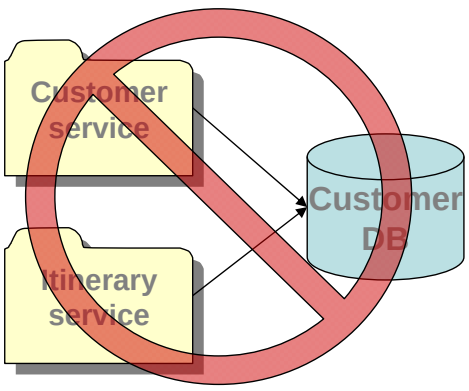
Provide interfaces for cross-enterprise integration. Thus they are of coarser granularity and have appropriate mechanisms for eg decoupling, security, billing or robustness.

Data-Centric Services

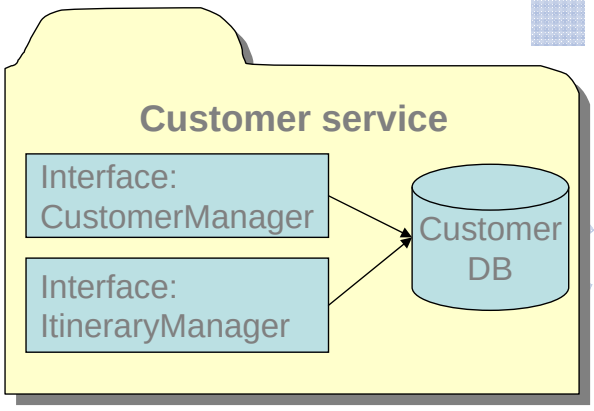
- Handle persistent data.
- Abstracts:
  - Storage and retrieval of data.
  - Locking mechanisms.
  - Transaction management.
- Similar to the data access layer of a traditional application architectures. But:
  - Traditional data access layer manages all data for the entire application (horizontal slicing).
  - A data centric service only deals with one major business entity (vertical slicing).

Data-Centric Services

- A data-centric service should strictly encapsulate its data entities.
  - Any other service that requires access to this data needs to use the service interface of the corresponding data-centric service.



a) Poor design: ownership of data unclear



b) Customer DB owned by customer service

Designing Data-Centric Services

- One of the most important tasks of an SOA architect is to identify relevant business entities that should be represented by data-centric services.
  - Similar to traditional business domain modelling.
  - Can use methods based on ER models or OO design, but cross service relationships are not allowed (as no cross-service navigation exists as in distributed object technology).
  - This means that complex data types used by services must be sufficiently self contained or must contain unique identifiers that enable them to relate to other complex data values.

Encapsulating Data and Behaviour

- Data-centric services should not only encapsulate access to the underlying persistent data, but also encapsulate logic and behaviour inherent to that data.
  - Similar benefits to object-orientation, but doesn't require OO language.
  - This should not however include business process logic or business (policy) rules which are likely to change more rapidly.

Logic-Centric Services

- Encapsulate algorithms for complex calculations or business rules (e.g. insurance pricing service)
- Stateless and functional in nature.

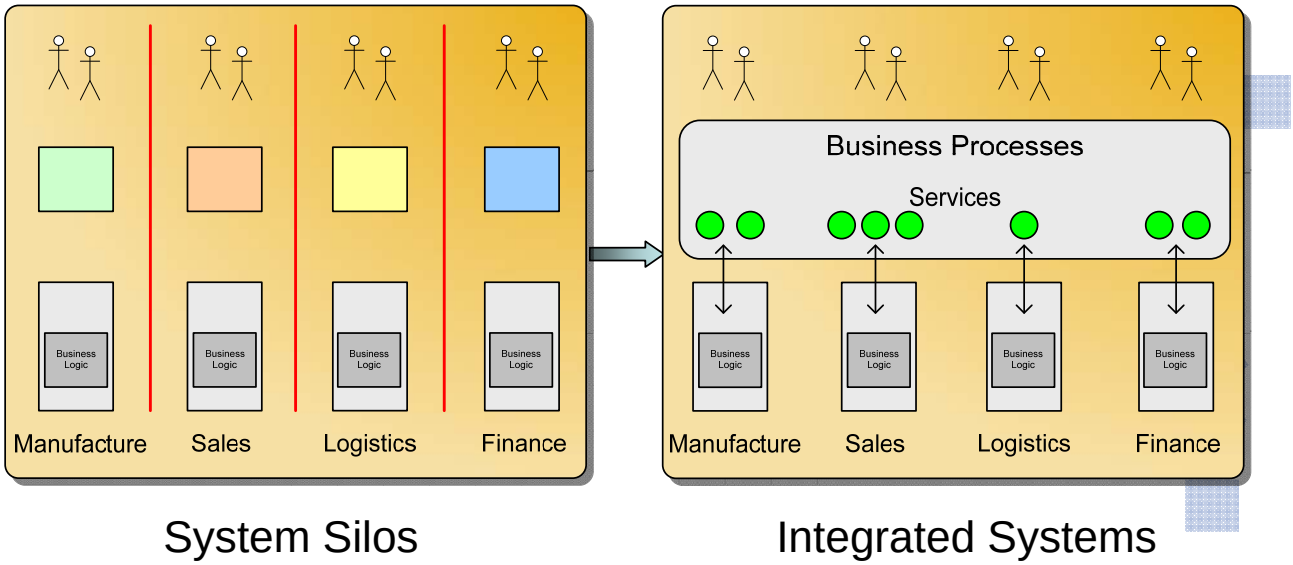
Intermediary Services

- Stateless mediators that bridge technological inconsistencies or design gaps in an architecture.
- Still provide a business oriented API.
- Include:
  - Technology Gateways
  - Adapters
  - Facades
  - Functionality-Adding Service

Process-Centric Services

- Encapsulate an organization's business processes.
  - Process logic is separated from application logic.
- They control and encapsulate the *state* of business processes.
  - Business processes may be long lived (eg months).
- The interaction between the client and provider may be arbitrarily complex
  - i.e. not limited to simple request/response patterns
  - however, patterns of interaction are normally known in advance and governed by a contract or formal understanding of what will be provided and under what conditions.

SOA & Business Process Management

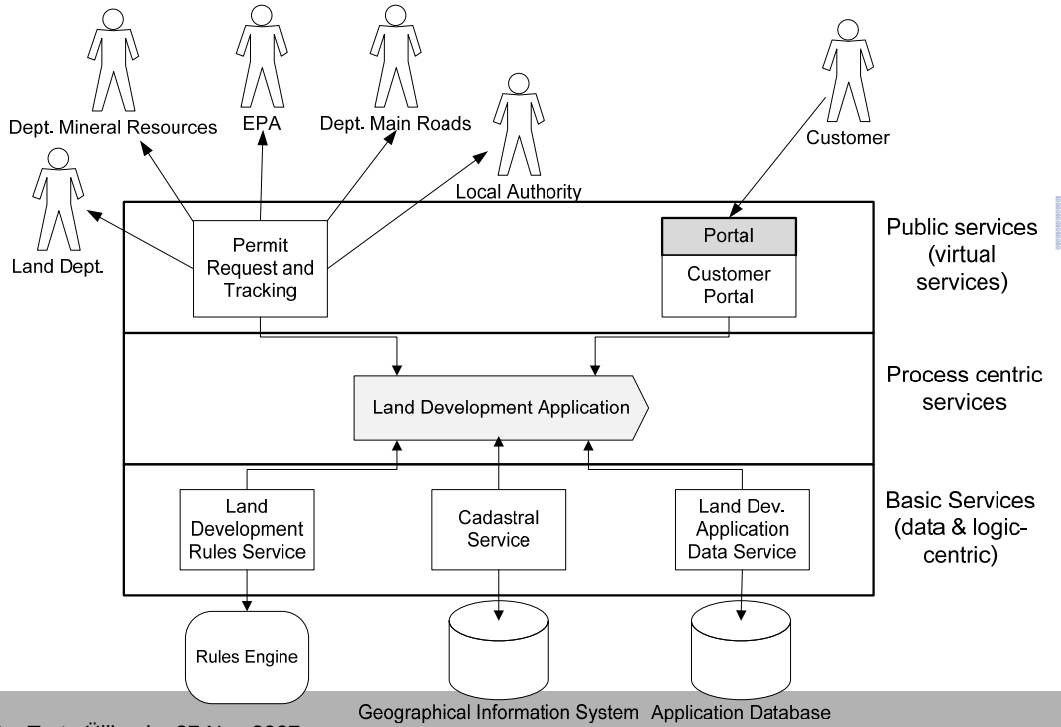


System Silos Integrated Systems

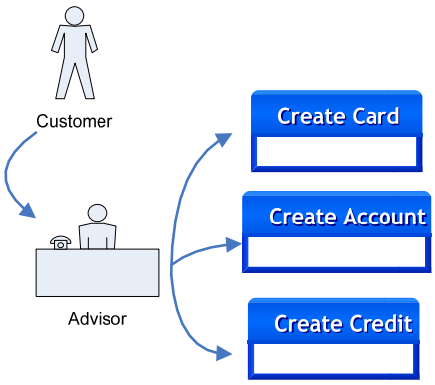
Public Enterprise Services

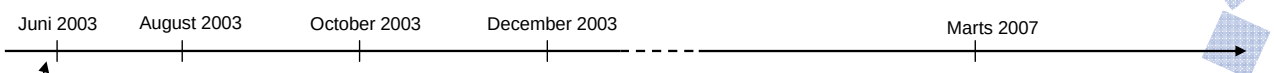
- Services that an enterprise offers to partners (B2B) and customers (B2C).
- These services have specific requirements:
  - Interfaces at the enterprise have the granularity of self-contained business documents.
  - Decouple business partners using asynchronous communication
  - Crossing the organization's borders implies higher security requirements, e.g. authentication, encryption and access control.
  - Billing for cross-enterprise services implies need for non-repudiation.
  - Regulated by Service Level Agreements.

SmartEDA: Integrated Land Development Application System



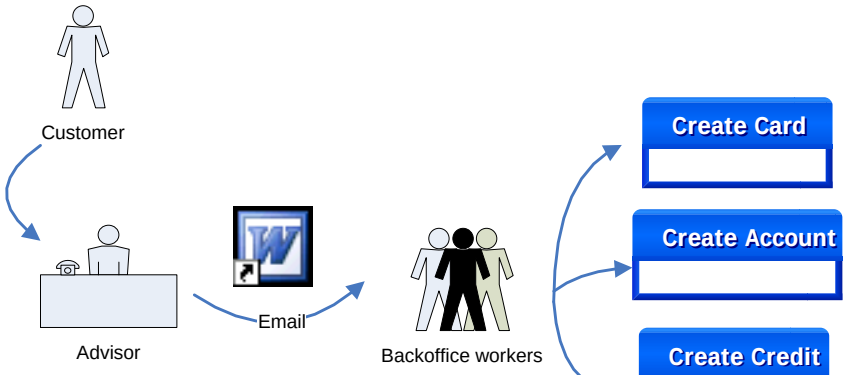
Danske Bank: Customer Package Process

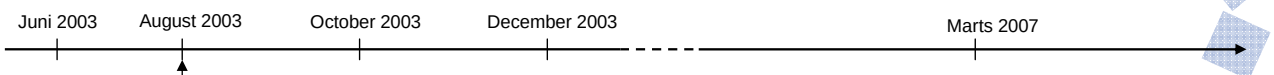




Introduction of Customer packages.
Word template to collect info

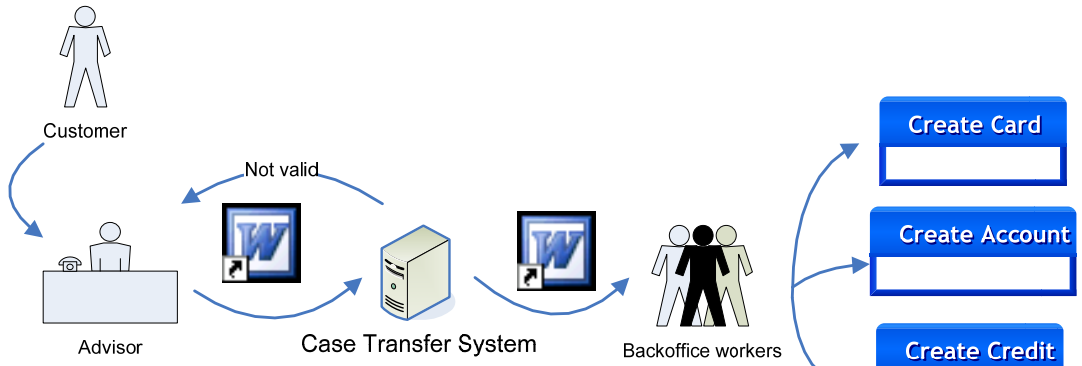
Danske Bank: Customer Package Process

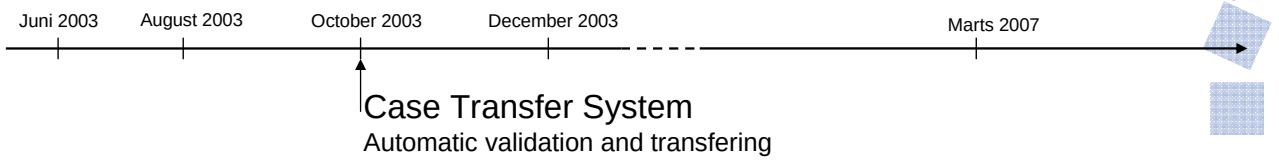




Backoffice group created
Handles the creation process

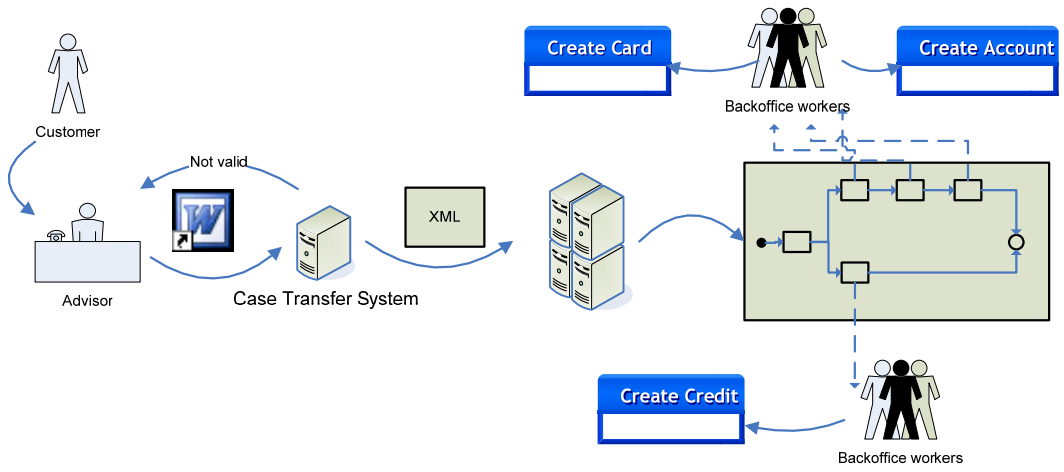
Danske Bank: Customer Package Process

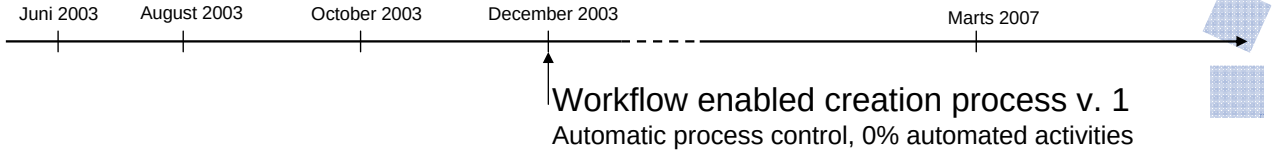




Case Transfer System
Automatic validation and transfering

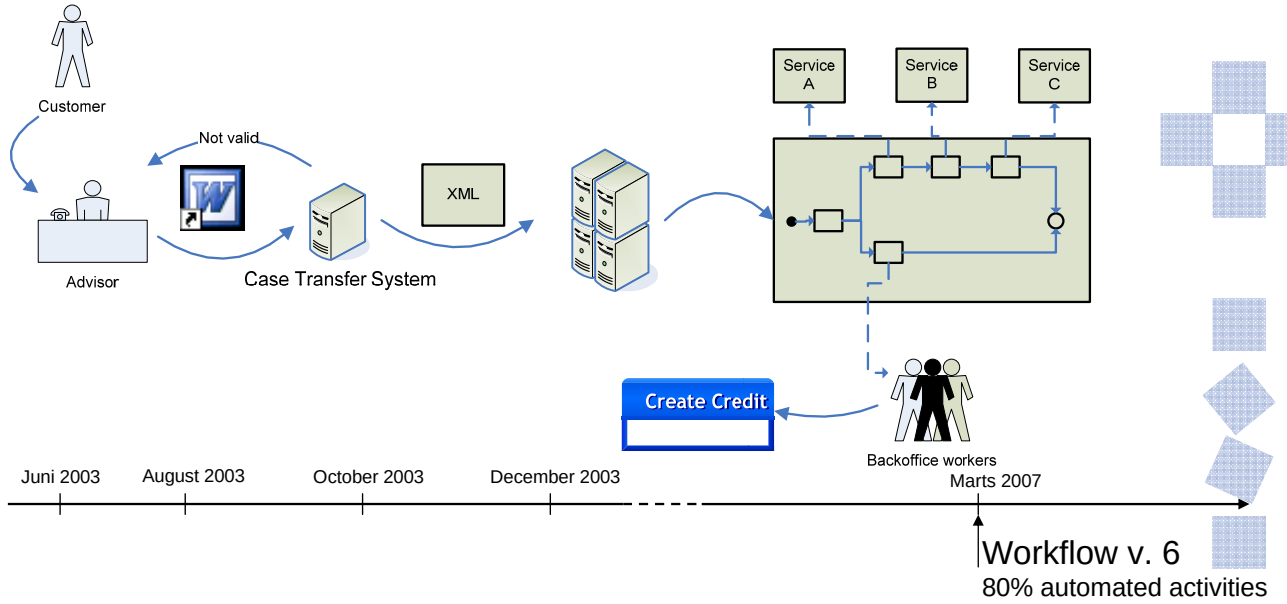
Danske Bank: Customer Package Process



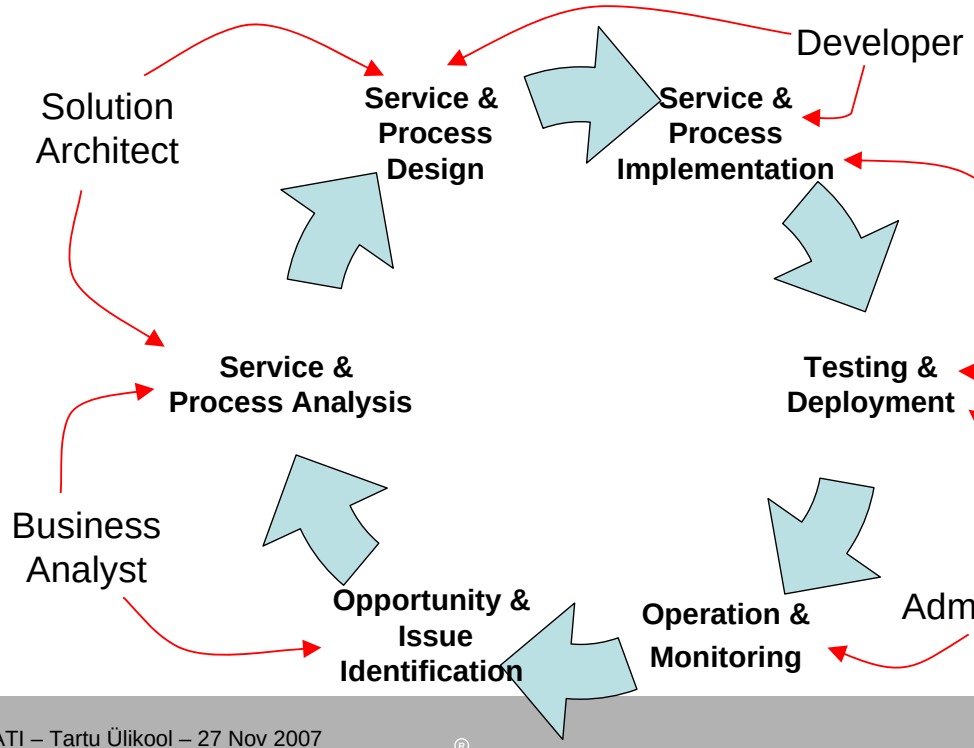


Workflow enabled creation process v. 1
Automatic process control, 0% automated activities

Danske Bank: Customer Package Process



Lifecycle and Roles in an SOA



Service Analysis and Design

Service Analysis: identification and definition of the business services that an organization provides or consumes, internally or externally.

Service Design: definition of a set of technical services to support the delivery of business services through IT.

Service Analysis & Design Methods

- Top-down capability-driven method
  - http://www.infoq.com/minibooks/enterprise-soa
- Bottom-up process-driven method
  - Thomas Erl: "Service-Oriented Architecture, Concepts, Technology, and Design", Prentice Hall, 2005
- Hybrid methods
  - www-128.ibm.com/developerworks/webservices/library/ws-soad1

Top-down Service Analysis

Pharmaceutical company "Pharmak" has a "natural medicines" business unit comprising four main areas:

- Sales:
  - contacts customer
  - receives order from customer
  - checks stock
  - requests for an order to be shipped - if item(s) on the order are available
- Manufacture:
  - makes item(s)
  - requests for an order to be shipped - after manufacturing the item(s)
- Logistics & Warehouse:
  - adds new item(s) into stock
  - requests an external company, or internal logistics, to deliver an order to a customer
  - receives supplies from suppliers

Top-down Service Analysis

- Finance:
  - prepares bill for customer
  - orders raw materials from suppliers
  - receives invoice from suppliers
  - prepares invoice for customer

The organization interacts with the following partners:

- Customer:

  organization which buys, and potentially sells on, jars
- Suppliers:

  manufacturers or wholesalers of components/raw materials needed to make jars
- Logistics Provider:

  used when the standard supply chain cannot cope with local demand, or to supply to global markets

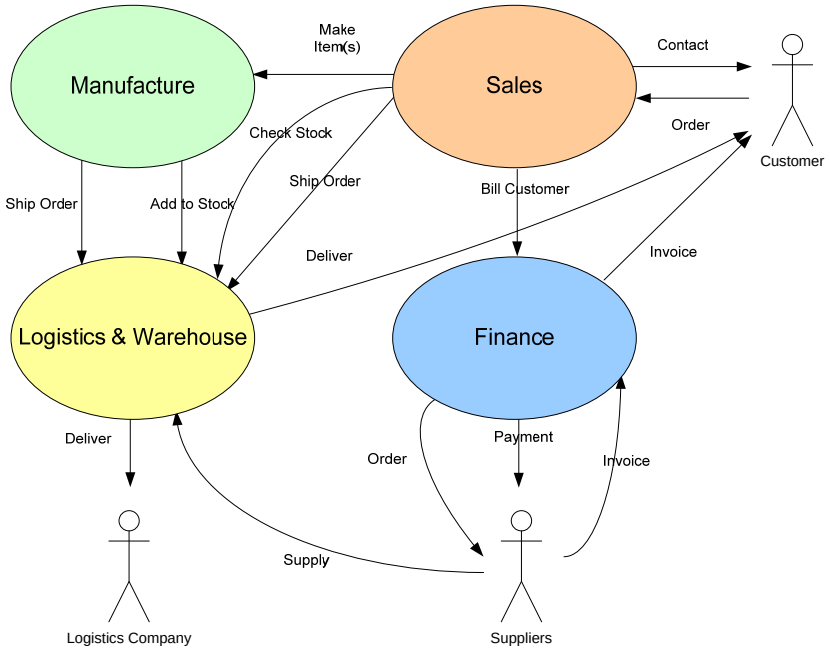
Top-down Service Analysis

Level 0 Architecture

What

Who

Why



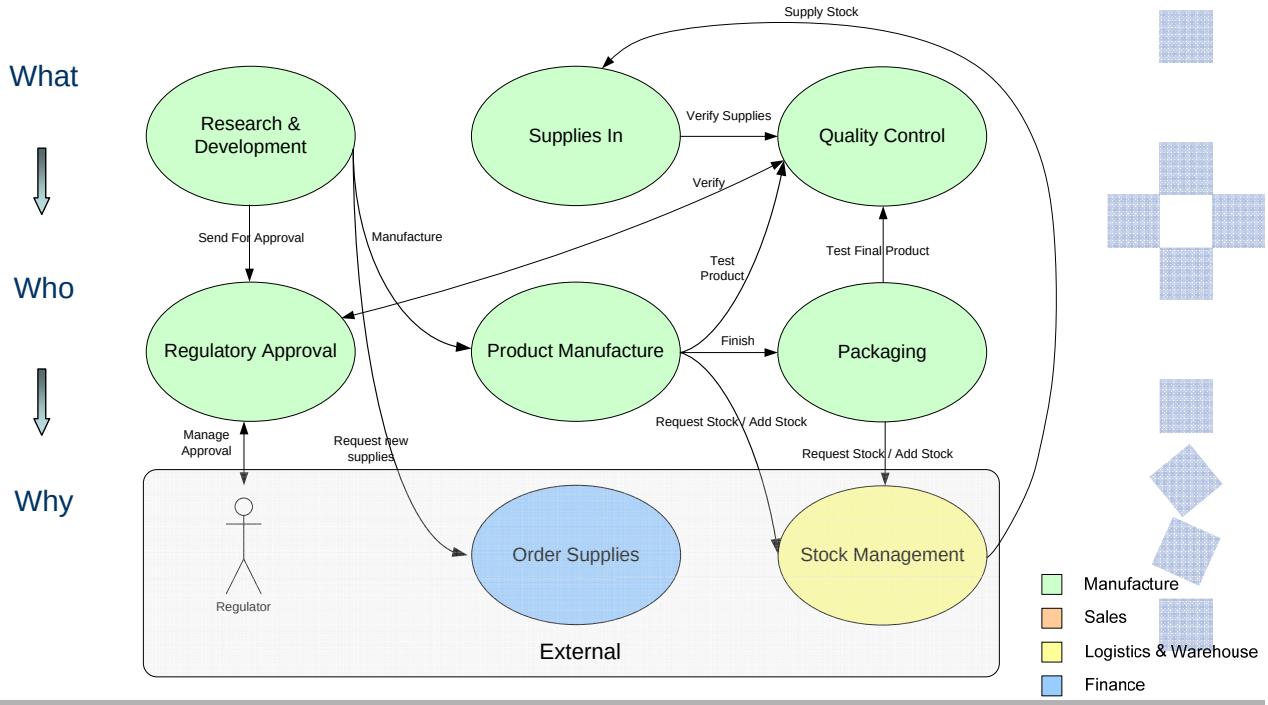
Top-down Service Analysis

Level 1 Architecture

- We reason in terms of “capabilities” (what can each area do?)
- Service analysis is carried out according to each Level 0 element identified before,
- Decomposition: the enclosing service confers behavior and management onto those at a lower level:
- As a rule of thumb, a maximum of 8 Level 1 services for each Level 0, with a normal amount around 4.

Top-down Service Analysis

Level 1 Architecture: Manufacture



Top-down Service Analysis

Level 2+ Architecture

Drilling down from Level 0 into lower abstraction level elements is a series of repetitions of the same steps.

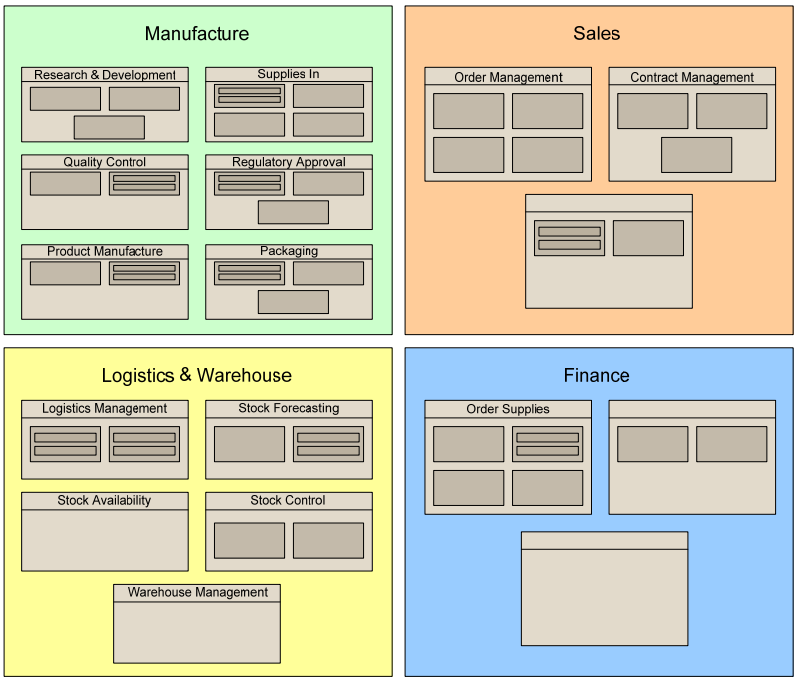
Purposes:

1. to delve deeper and understand the problem domain more,
2. to identify:

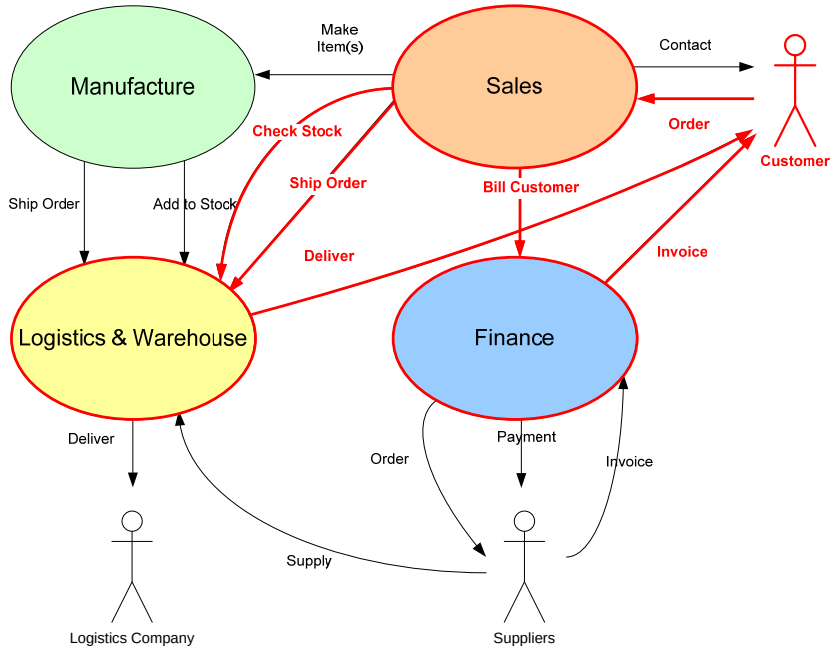
- Virtual Services
- Support Services
- Shared Services

Top-down Service Analysis

The Service Map



Top-down Service SOA – Design Phase

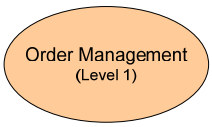


Top-down Service Design

Services involved:

- Sales:
  - Order Management:
    - Make Quote
    - Process Order
    - Cancel Order

    } Order
    - Check Availability — Check Stock
    - Request Shipment — Ship Order
    - Request Bill — Bill Customer
- Logistics & Warehouse:
  - Stock Availability:
    - Provide Availability — Check Stock
  - Warehouse Management:
    - Stock Out — Ship Order
    - Deliver



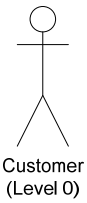
Top-down Service Design

- Finance:
  - Customer Management:
    - Make Bill — Bill Customer
    - Invoice

Services involved in the Customer:

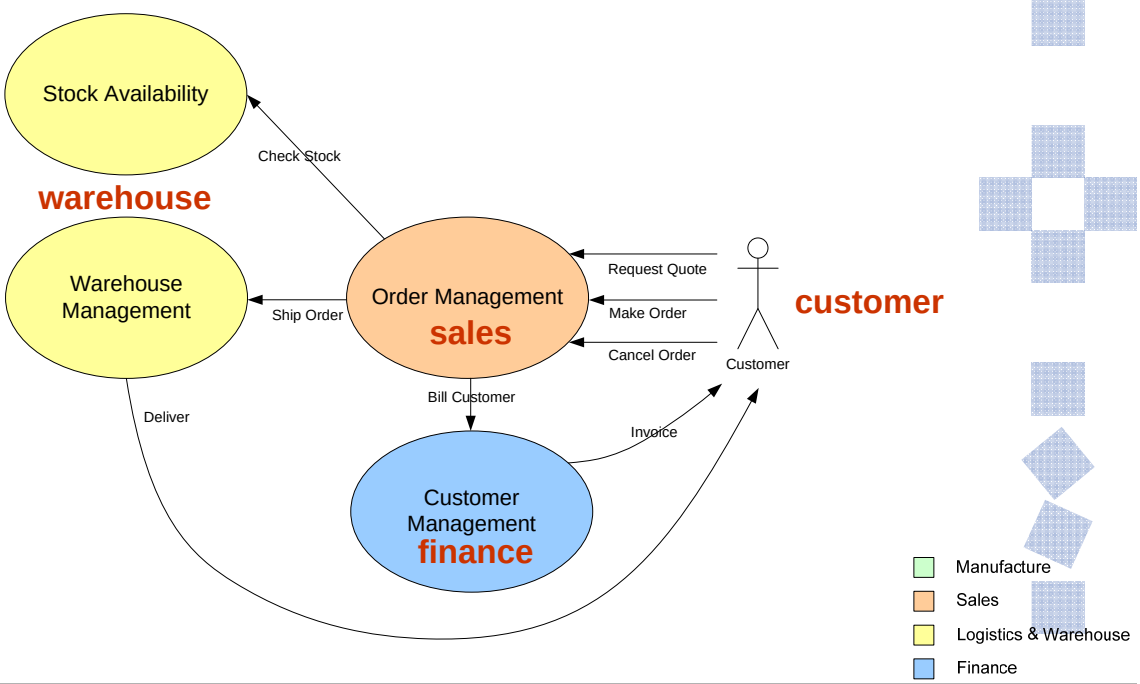
- Request Quote
- Place Order
- Cancel Order

} Order

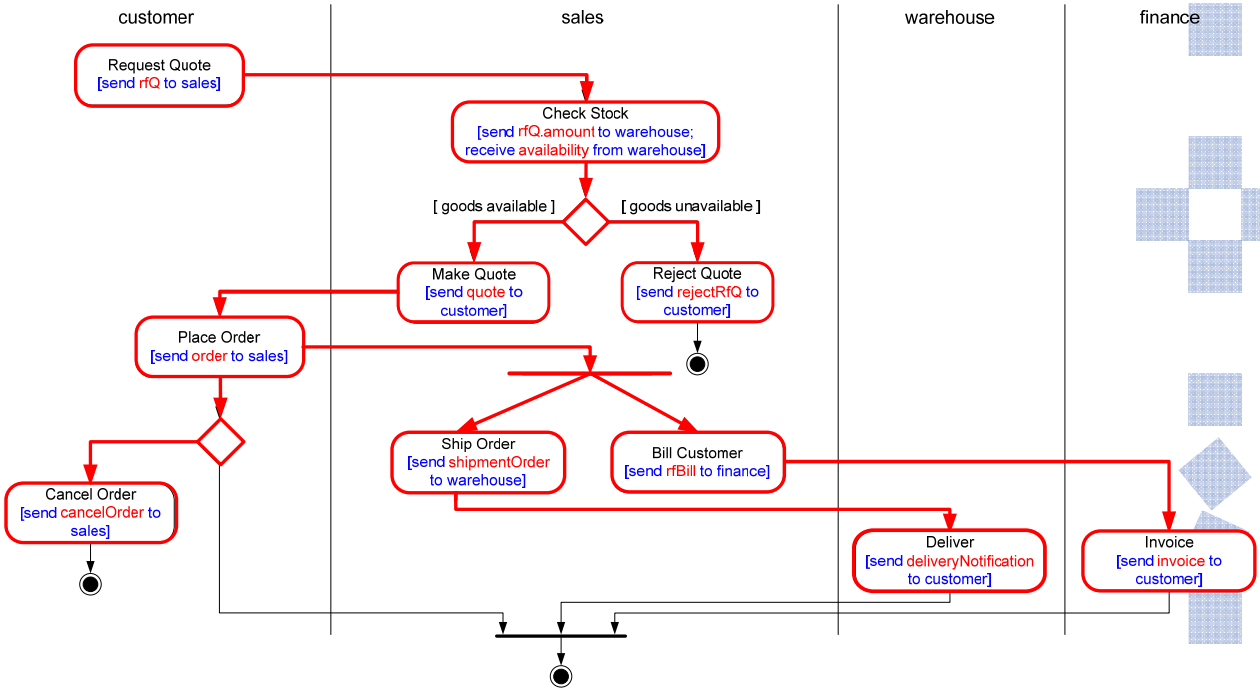


Top-down Service Design

Let's refine the interactions…and define the roles:



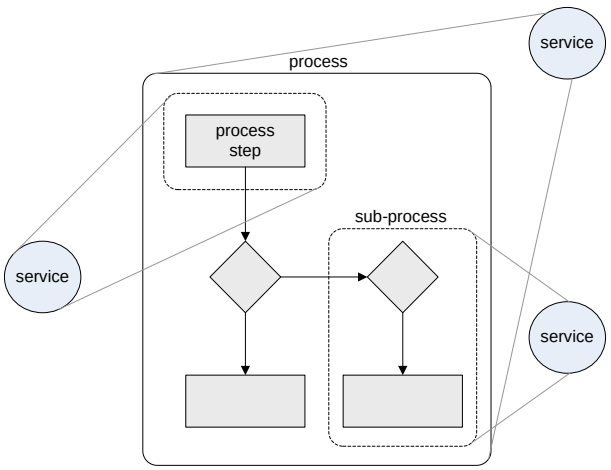
Top-down Service Design



Bottom-up Process-driven Service Analysis

Business Process Models can be used as the source which to identify services:

- a thorough knowledge of the underlying workflow logic is required,
- the scope of the identified services may vary:



Bottom-up Process-driven Service Analysis

After loosing its unique client “TLS”, “RailCo Ltd.” needs to find new customers.

This leads the company to build a standardized set of services, generic enough to facilitate interactions with multiple clients.

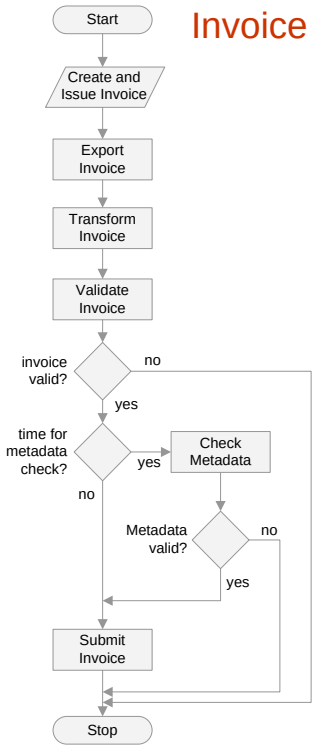
To pursue this goal, RailCo Ltd. decides to overhaul its existing environment in favor of an SOA.

The service analysis is conducted over two *internal* business processes, pitched to work with the B2B solution of TLS:

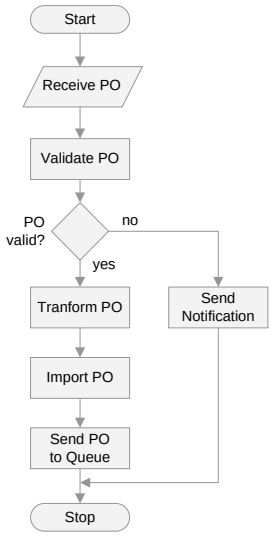
- Invoice Submission Process: sends an invoice to TLS, uses the Invoice Submission Web Service.
- Order Fulfillment Process: accepts and processes Purchase Orders from TLS, uses the Order Fulfillment Web Service.

Bottom-up Process-driven Service Analysis

Invoice Submission

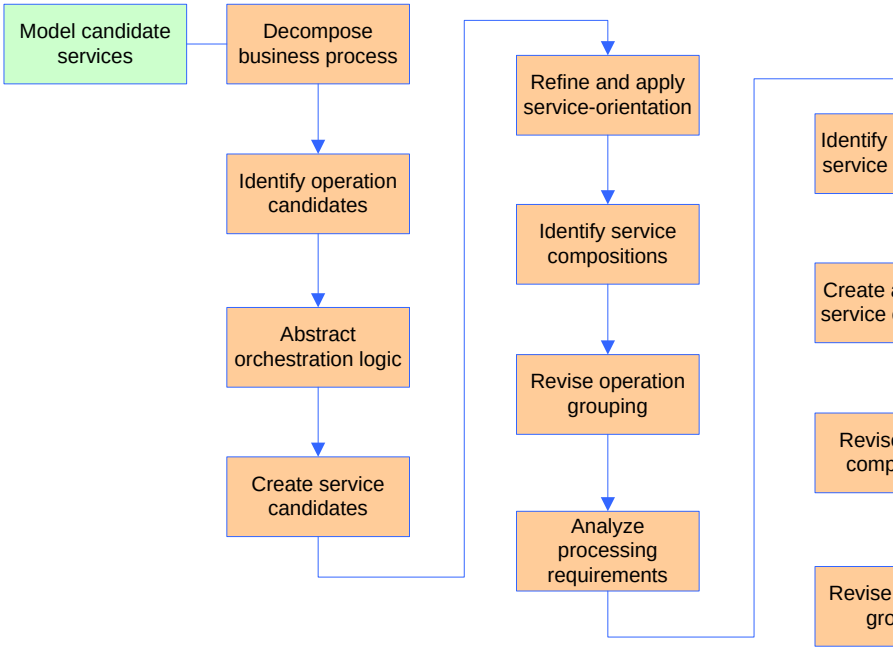


Order Fulfillment



Bottom-up Process-driven Service Analysis

Applying the framework: Step 3 - Model candidate services



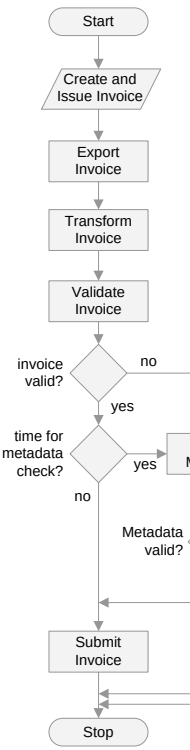
Bottom-up Process-driven Service Analysis

1 – Decompose the business process

Break down the workflow logic in the “most granular” representation of process steps.

Invoice Submission Process

- Create electronic invoice,
- Issue electronic invoice,
- Export electronic invoice to network folder,
- Poll network folder,
- Retrieve electronic invoice,
- Transform electronic invoice to XML document,
- Check validity of invoice document. If valid, end,
- Check if it is time to verify TLS metadata,
- If required, perform metadata check. If the check fails, end.



Bottom-up Process-driven Service Analysis

2 – Identify business service operation candidates

Some steps can be easily identified as not belonging to the potential logic to be encapsulated in a service candidate (e.g. activities performed manually or by some legacy logic).

Invoice Submission Process

~~Create electronic invoice,~~	Manual step (accounting clerk)
~~Issue electronic invoice,~~	Manual step (accounting clerk)
Export electronic invoice to network folder,	Custom developed component (legacy logic)
Poll network folder,	Performed by a custom developed component
Retrieve electronic invoice,	Performed by a custom developed component
Transform electronic invoice to XML document,	Performed by a custom developed component
Check validity of invoice document. If valid, end,	Performed by the Invoice Submission WS
Check if it is time to verify TLS metadata,	Performed by the Invoice Submission WS
If required, perform metadata check. If the check fails, end.	Performed by the Invoice Submission WS

Could become part of a generic service candidate.

Could become a separate service candidate.

Bottom-up Process-driven Service Analysis

3 – Abstract orchestration logic

Identify the parts of the processing logic that this layer would potentially abstract (e.g. business rules, conditional / exception / sequence logic).

The workflow logic of separate process service candidates, derived from the Invoice Submission and Order Fulfillment processes would include the following conditions:

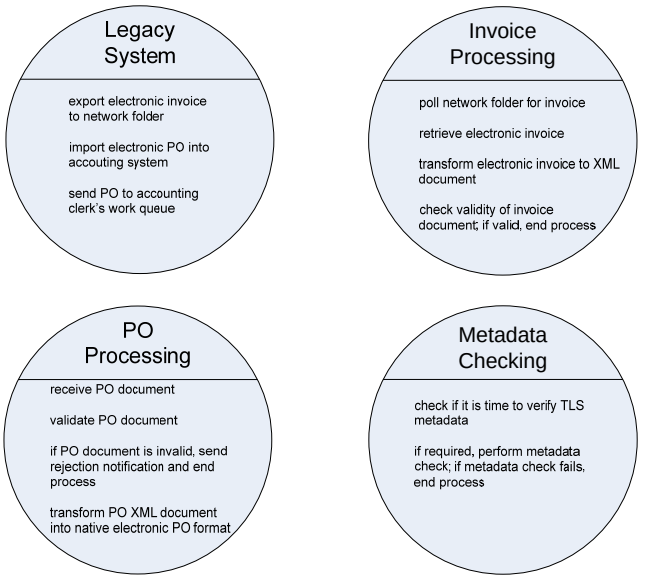
- if the invoice document is valid, proceed with the metadata check,
- else, end process,
- if the interval period for performing a metadata check has completed, perform the metadata check,
- else, skip the metadata check.

- if the PO document is valid, transform the PO document,
- else, end process.

Bottom-up Process-driven Service Analysis

4 – Create business service candidates

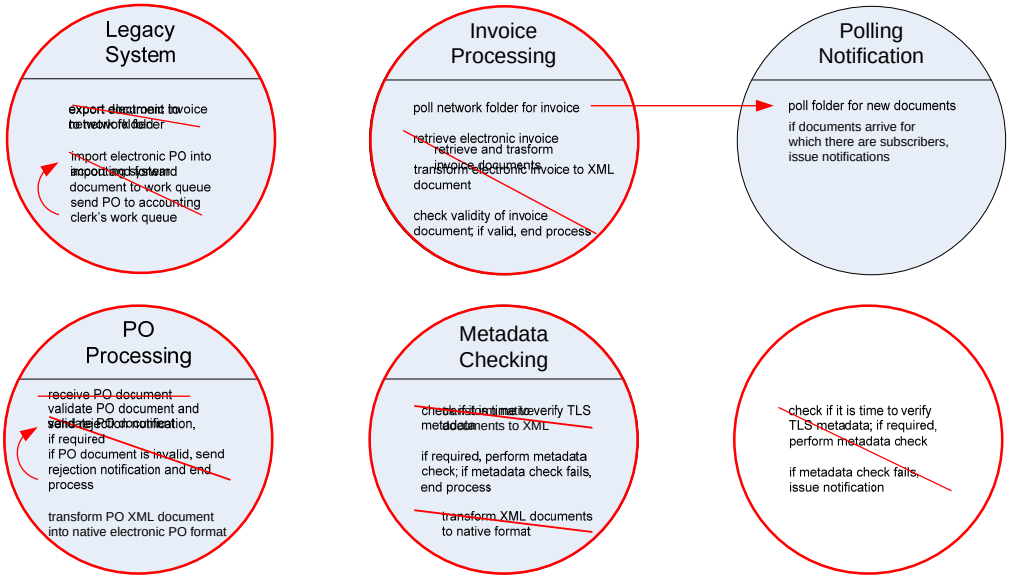
- The identified steps are grouped by logical context, with each group representing a service candidate.
- The context depends on the type of the business services chosen.



Bottom-up Process-driven Service Analysis

5 – Refine and apply principles of service-orientation

Refine the candidates according to the principles of reusability and autonomy.

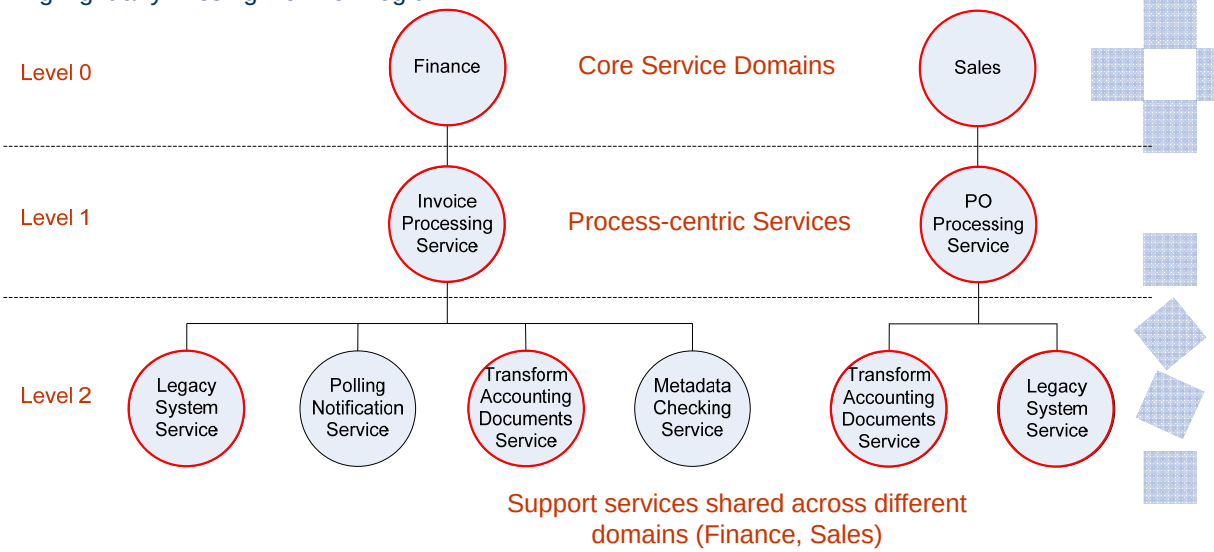


Bottom-up Process-driven Service Analysis

6 – Identify candidate service compositions

Identify the set of most common scenarios that can take place, in order to:

- evaluate the appropriateness of the candidate contexts,
- identify potential service composition,
- highlight any missing workflow logic.



Bottom-up Process-driven Service Analysis

The reverse of the medal: *Percolating Processes*

Organizations start with a detailed Process Map and then try to fit this into a SOA:

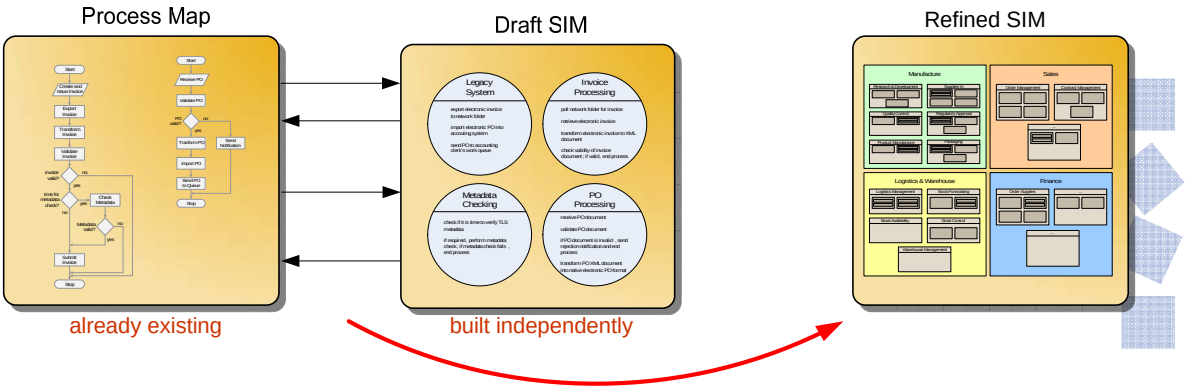
- processes become the dominant feature: POA rather than SOA,
- task-centric business services (Level 1) are tightly bound to the context of a single process → become difficult or impossible to change,
- fine grained services (Level 2+) proliferate and become difficult to manage.

Effects

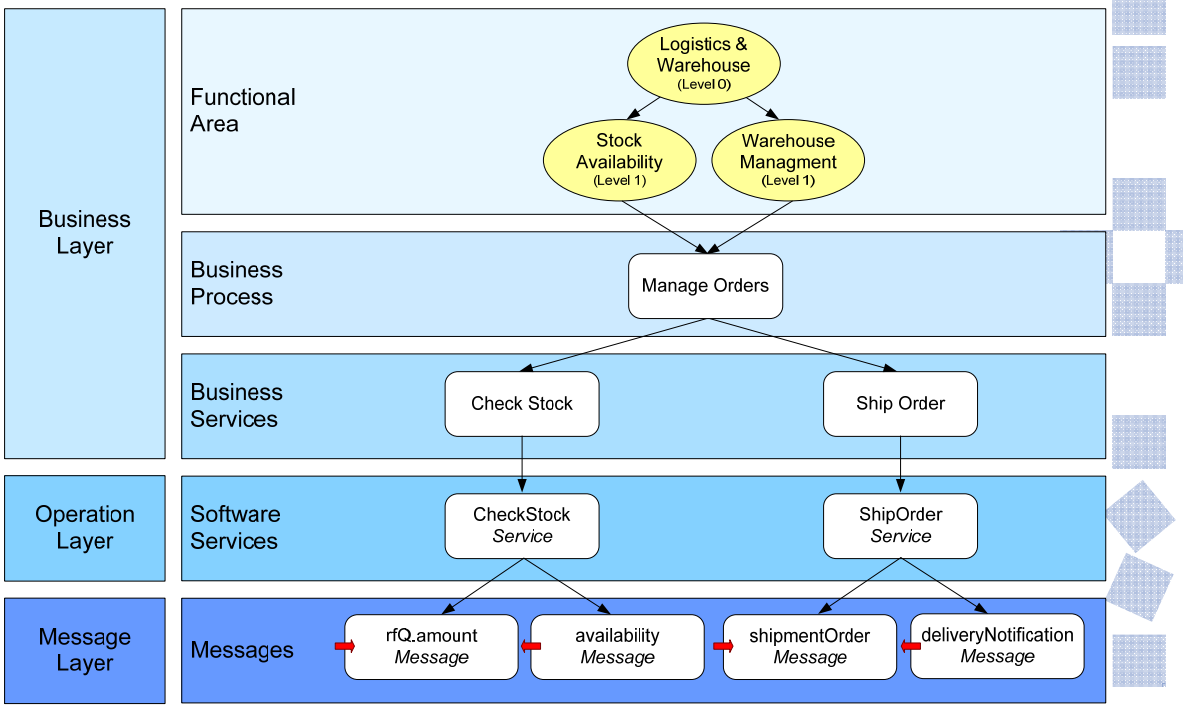
- the system slowly or quickly stagnates,
- new solutions are built on top of the existing ones, due to the lack of reusability (process-oriented system treated as legacy application).

Service Analysis and Design – Meet-in-the-Middle

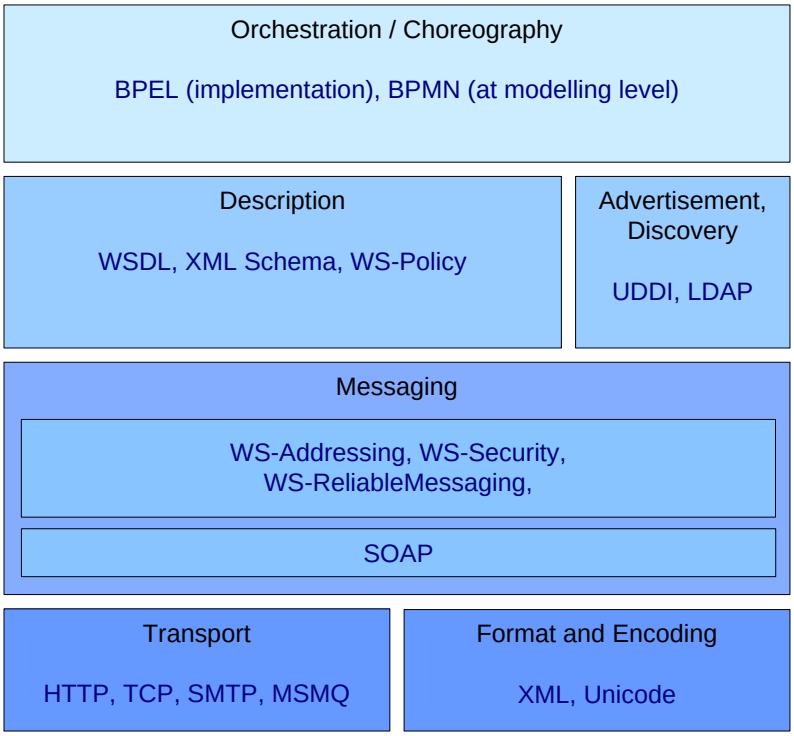
1. Create the Service Interaction Map (SIM) independently of the Process Map: this provides the structure for:
   - breaking down the processes,
   - creating a clear hierarchy of use.
2. Overlay the SIM onto the Process Map, to understand potential cuts.
3. Refactor the current solution by attacking the major inflexibilities.



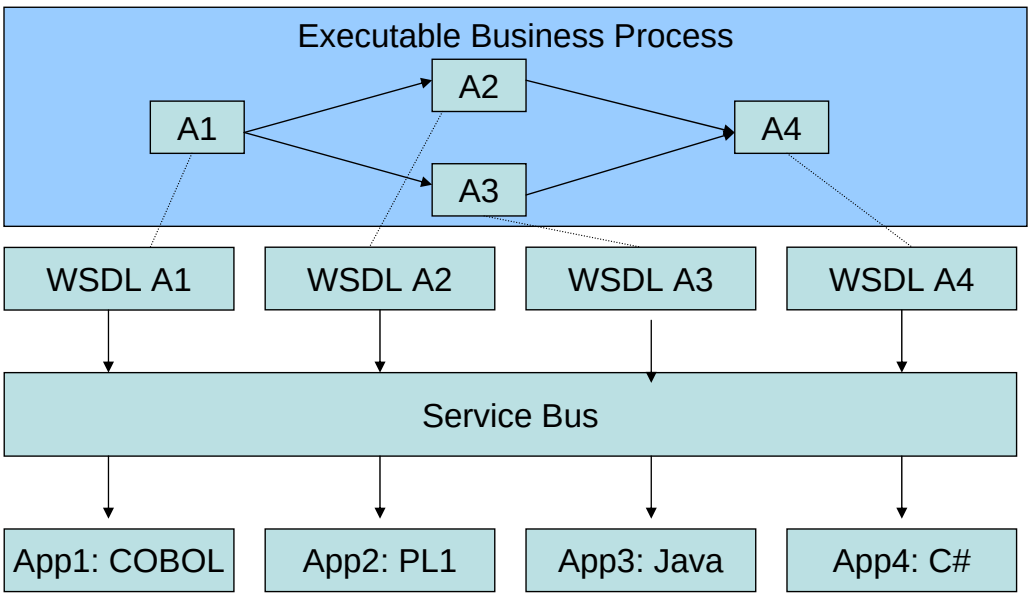
Synthesis: Service Definition Layers



Web Service Technology Standards



Implementing Process-Centric Services



Process-centric Services: BPEL

Business Process Execution Language (BPEL)

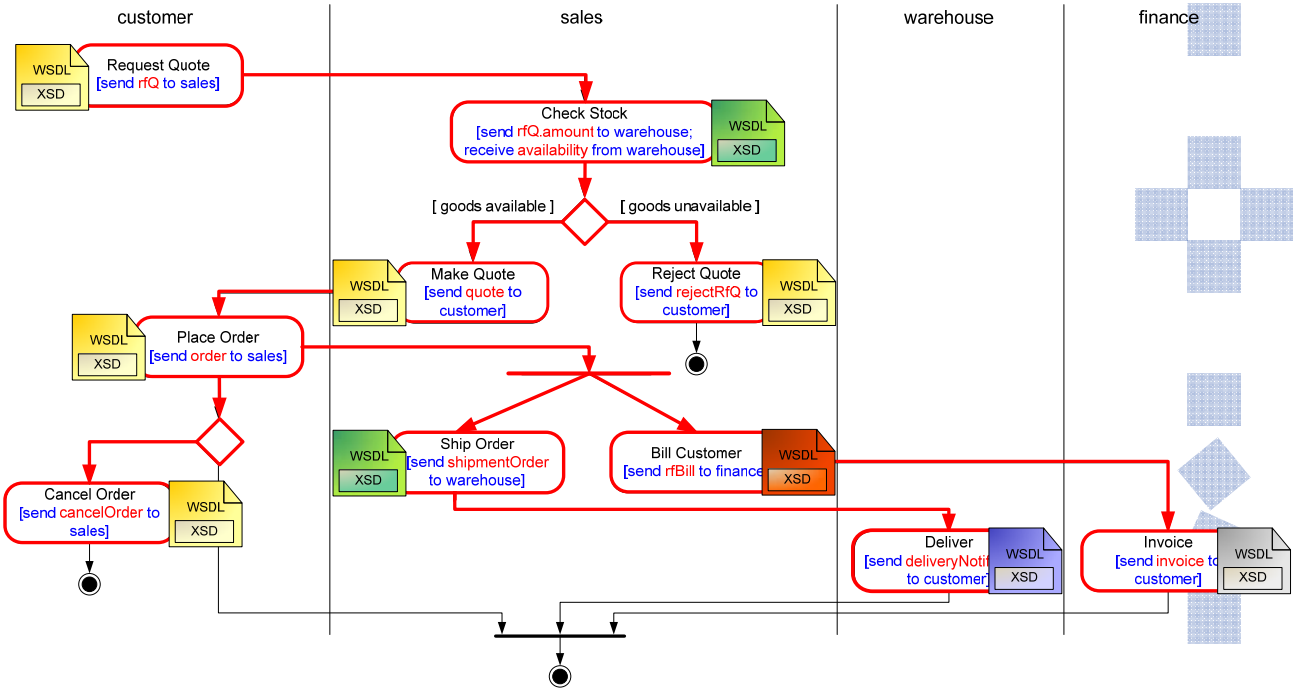
- Supports description of behavioural interfaces and orchestrations:
  - executable processes model an executable private workflow (orchestrations);
  - abstract processes specify a public message exchange (behavioral interfaces);
- Imperative programming language: scoped variables, assignment, sequence, while, switch, exception handlers;
- Constructs specific to WS programming:
  - XML variable typing with XPath/XQuery/XSLT expressions;
  - primitive receive and send actions (with dynamic references);
  - parallelism and synchronisation: flow and control links;
  - event-action rules (event handlers);
  - nested transactions with compensation (compensation handlers);
  - In BPEL 2.0: multiple concurrent executions of a block of code;

Process-centric Services: BPEL

Business Process Execution Language (BPEL) (cnd)

- Layered on top of WSDL:
  - every process is exposed as a web service (or set of web services) using WSDL documents, which describe the process's structural interface (for WS Composition);
  - WSDL references specify calls to external services;
  - WSDL types are used to describe persistent information (variables);
  - independent of WSDL implementation (i.e. binding and service) – relies only on the abstract part.

BPEL Example: Order Management – artifacts



SOA Trends

Service Marketplaces & Software as a Service

- Service marketplaces controlled by a single entity (e.g. SalesForce AppExchange, StrikeIron, SOATrader, Amazon Web Services).
- Open service communities built around services provided by a single entity (e.g. Google mashups).
- E-Government service marketplaces (e.g. BizDex, DirectGov).
- Mobile Service Marketplaces (e.g. Intel Digital Communities Initiative, Streamspin.com)