Internet-based Self-Services from Analysis and Design to Deployment

Jorge Cardoso
Dept. Engenharia Informatica/CISUC
University of Coimbra
Coimbra, Portugal
jcardoso@dei.uc.pt

John A. Miller
Computer Science Department
University of Georgia
Athens, GA, USA
jam@cs.uga.edu

IEEE SE 2012 1st International Conference on Services Economics June 24-29 2012, Hyatt Regency Waikiki Resort and Spa, Honolulu, Hawaii, USA

Roadmap

- Motivation, objectives, and findings
- SEASIDE
  - Service architecture
  - Service analysis
  - Service design
  - Service implementation
  - Service deployment
- Conclusions

Motivation

- Past research in the field of services
  1. Software and IT perspective
  2. Business and marketing
  3. Service design

- Our goal
  - Bridge **1-3**
  - **Investigate a Systematic approach for Internet-based Self-Service (ISS) development**

What is an Internet-based Self-Service (ISS)

- Subclass of real-world services
- No direct service employee
- Driven by self-service technologies
- The Internet as a delivery channel

Self-ticket purchasing using the Internet is an example of an Internet-based self-service

Objectives

- Systematic approach
  - Guiding framework
  - Support analysis, design, implementation and deployment
- Improve traceability and consistency
- Reduce complexity and development costs

SEASIDE

- Systematic ISS development
- Rely on:
  - Enterprise architectures (EA),
  - Model-driven development (MDD),
  - Model-view-controller pattern (MVC), and
  - Platform as a service (PaaS)

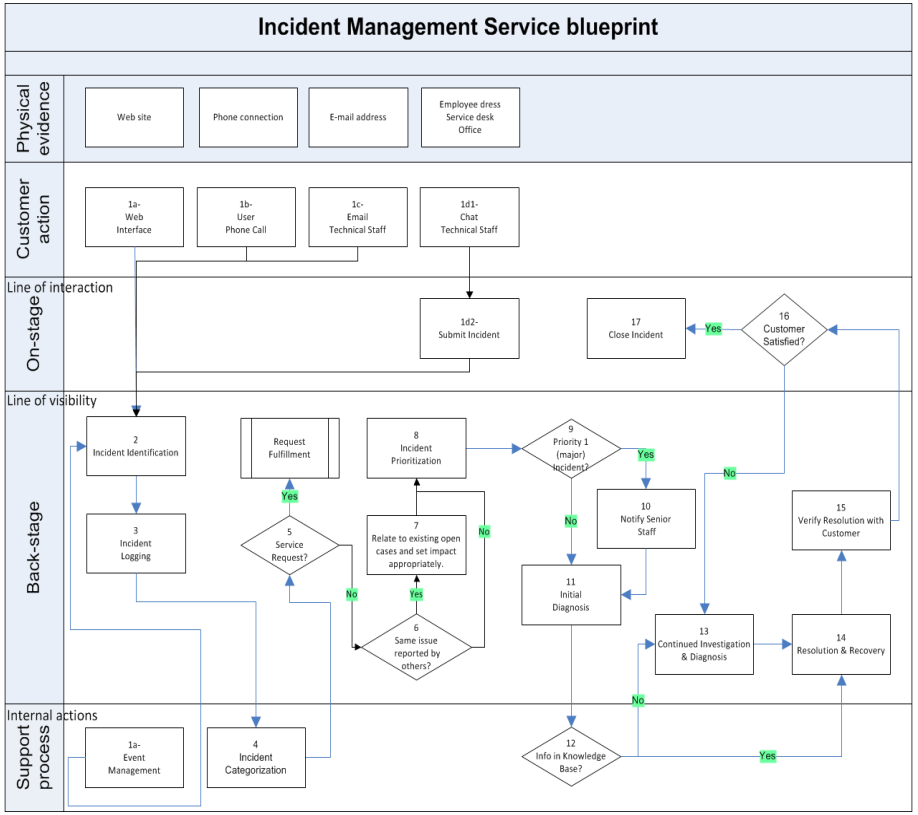
SEASIDE = **S**yst**E**m**A**tic **S**erv**I**ce **DE**velopment

Findings

- SEASIDE
  - Increases traceability (EA, MDD)
  - Increases consistency of code (MDD, MVC)
  - Reduces complexity (EA, MDD, MVC)
  - Lowers deployment costs (PaaS)

ITIL IMS Use Case

- Incident Management Service (IMS)
- From ITIL
- Blueprint >>
- Resolve incidents
- Disks errors
- Printers not working

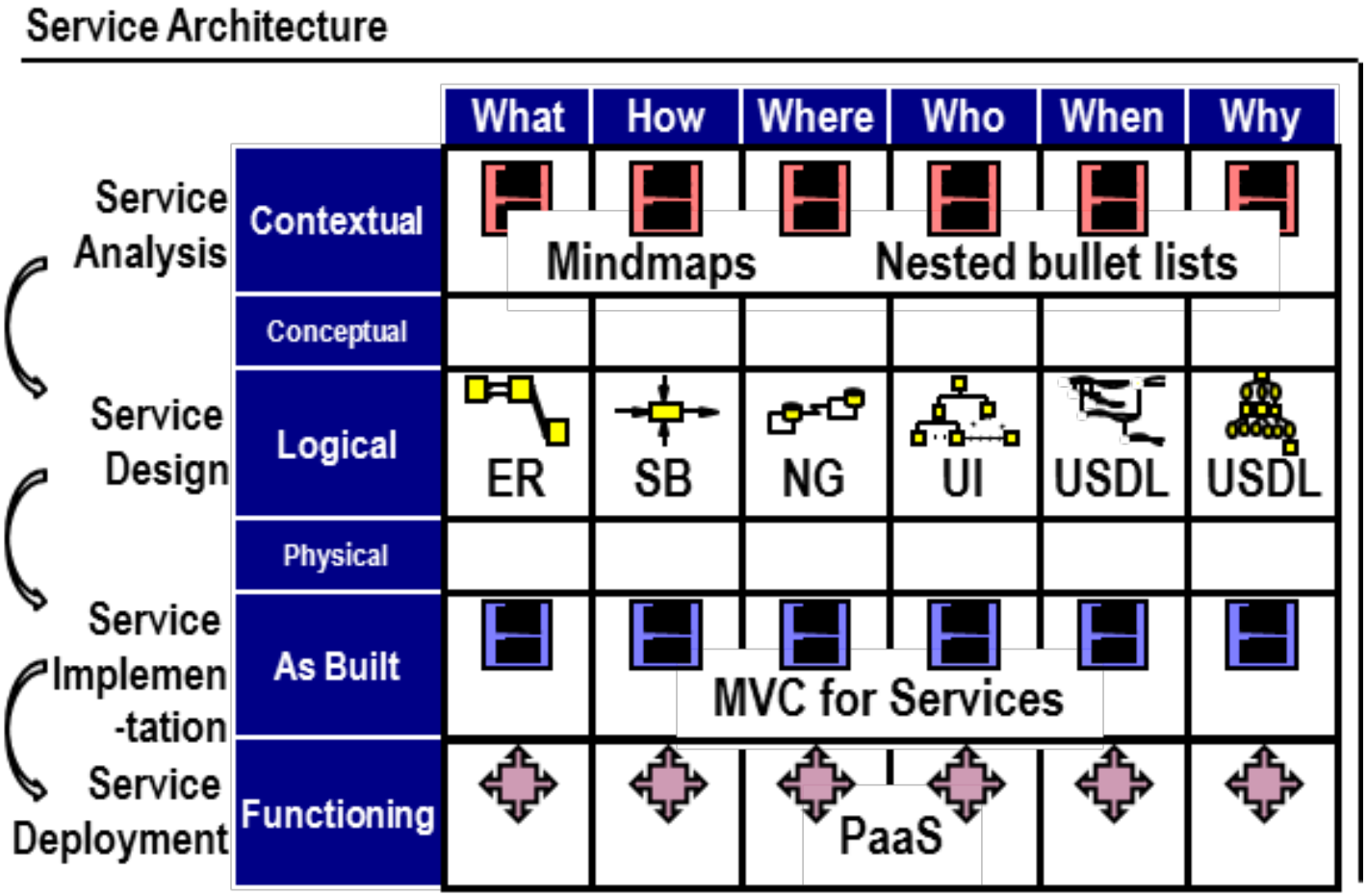


- ITIL = Information Technology Infrastructure Library


SEASIDE

The
service architecture
is a blueprint which
captures artifacts
produced during
ISS development

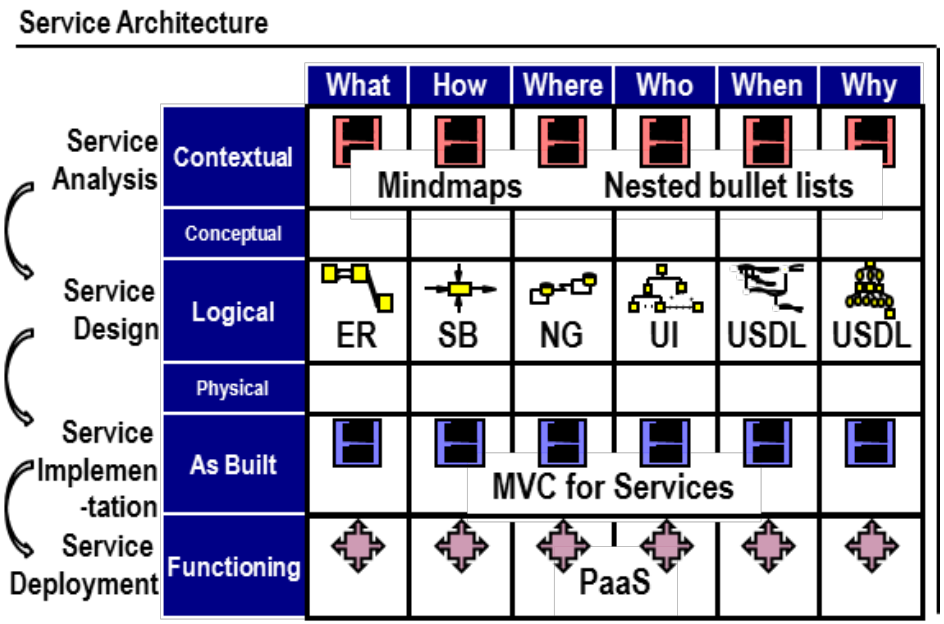
SEASIDE: Service architecture



Zachman enterprise architecture

SEASIDE: Service architecture

- Adoption of an enterprise architecture
- Organize service development
- Control completeness
  - models, interfaces and linkings



Zachman enterprise architecture

SEASIDE: Service architecture

- Service analysis
  - Contextual abstraction (mindmaps, nested bullet lists, etc.)
- Service design
  - Logical abstraction (models, MDD)
- Service implementation
  - *As build* abstraction (MVC)
- Service deployment
  - Functioning abstraction (PaaS)

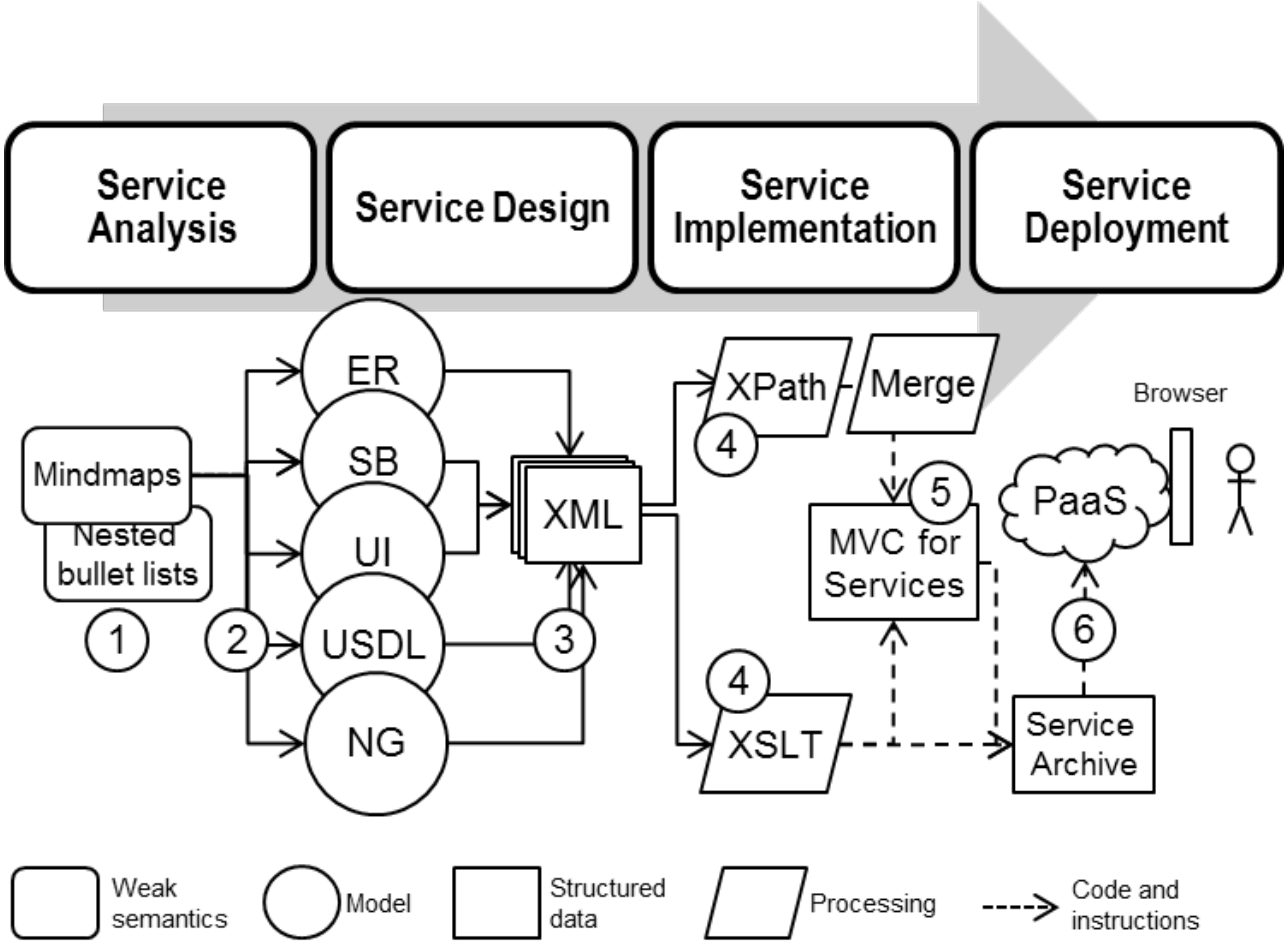
SEASIDE

systematic

development drives

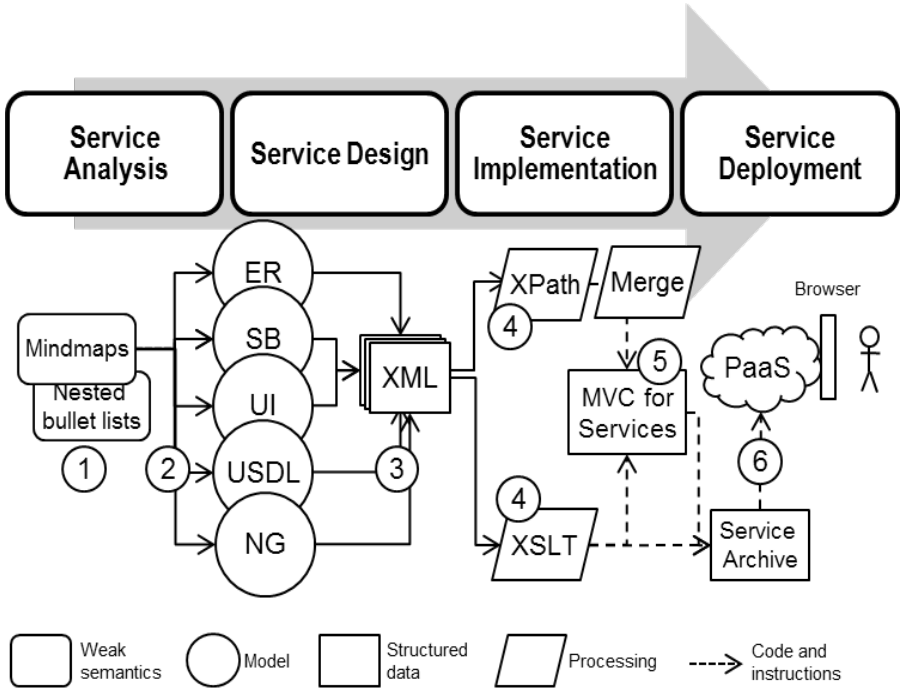
service analysis, design, implementation and deployment

SEASIDE: Systematic development



SEASIDE: Systematic development

- Steps 1-2)
  - **Construct** service models based on the semantics captured in the analysis phase.
- Steps 3-4)
  - Models are **serialized** to XML.
  - Code/instructions are **generated** from models using XSLT.
- Step 5)
  - Code/instructions are **organized** using MVC structure customized for services.
  - The service is automatically **deployed** into a commercial PaaS platform.



Service analysis

- Use weak semantics to identify ISS main concepts

- ITIL Incident Management Service case study

1) **Incident**. Unplanned interruption of an IT service.
    a) **Priority**. How quickly the service should address the incident.
        i) **Impact**. The effect on the business.
        ii) **Urgency**. The need for a fast resolution.
    b) **Supervisor**. The responsible actor.
2) **Solution**. Steps to solve the incident.
3) **Customer**. The actor which submitted the incident.
4) . . . .

Service design

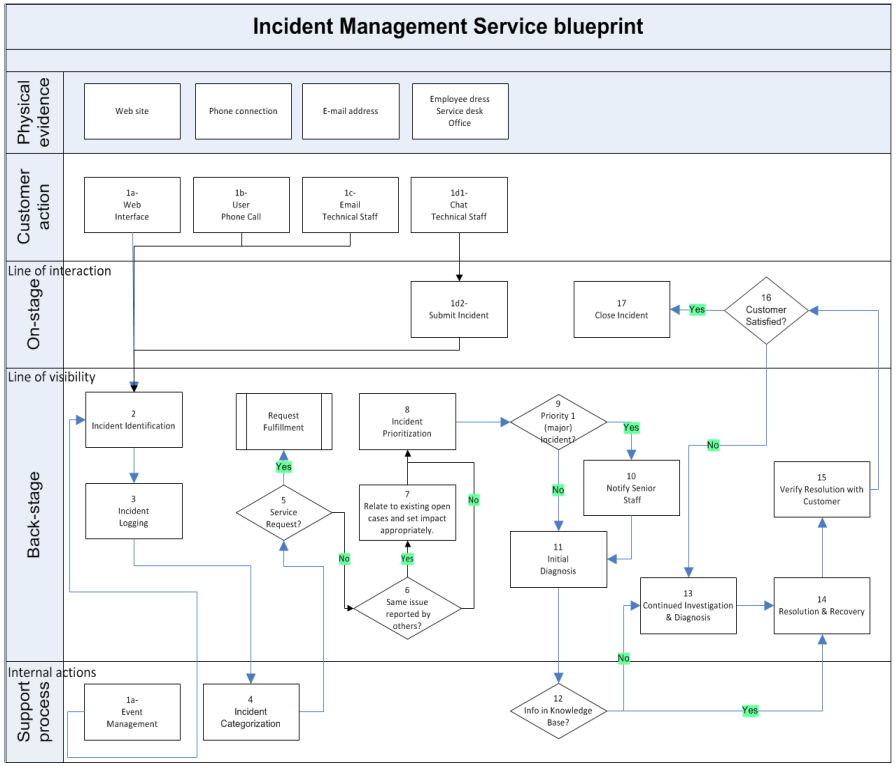
- Aggregates several models according to the service architecture

- ITIL Incident Management Service use case
  - Entity-Rel. models (ER) for modeling data (’what’)
  - Service Blueprinting (SB) to model functions (’how’)
  - Network Graphs (NG) to model locations (‘where’)
  - Low fidelity prototypes to model user interfaces (UI) (’who’)
  - USDL to model schedules and motivations (’when’, ’why’)

Example: Service blueprint

- BPMN to design service blueprints
- Mapping between BPMN and blueprints
- Transform BPMN into a MVC pattern

1) Sparx Systems' Enterprise Architect
2) Bonita BPMN workflow editor

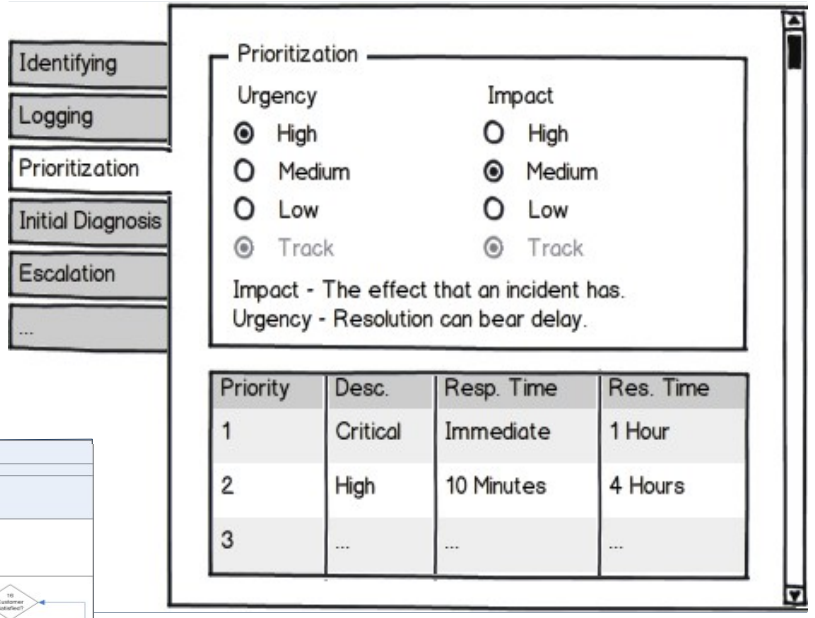


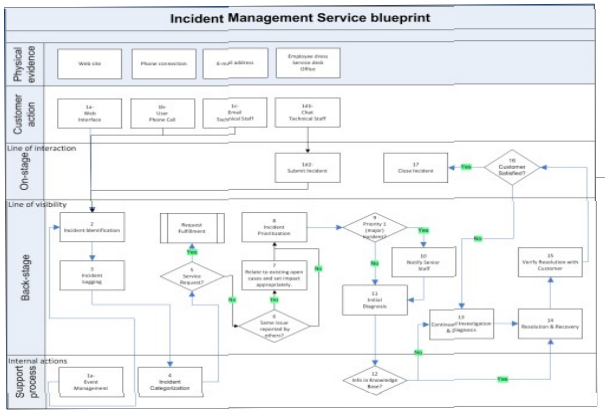
XML serialization

Example: UI

- Blueprint tasks are associated with a UI view
- Low fidelity prototypes are created manually

Balsamiq mockups

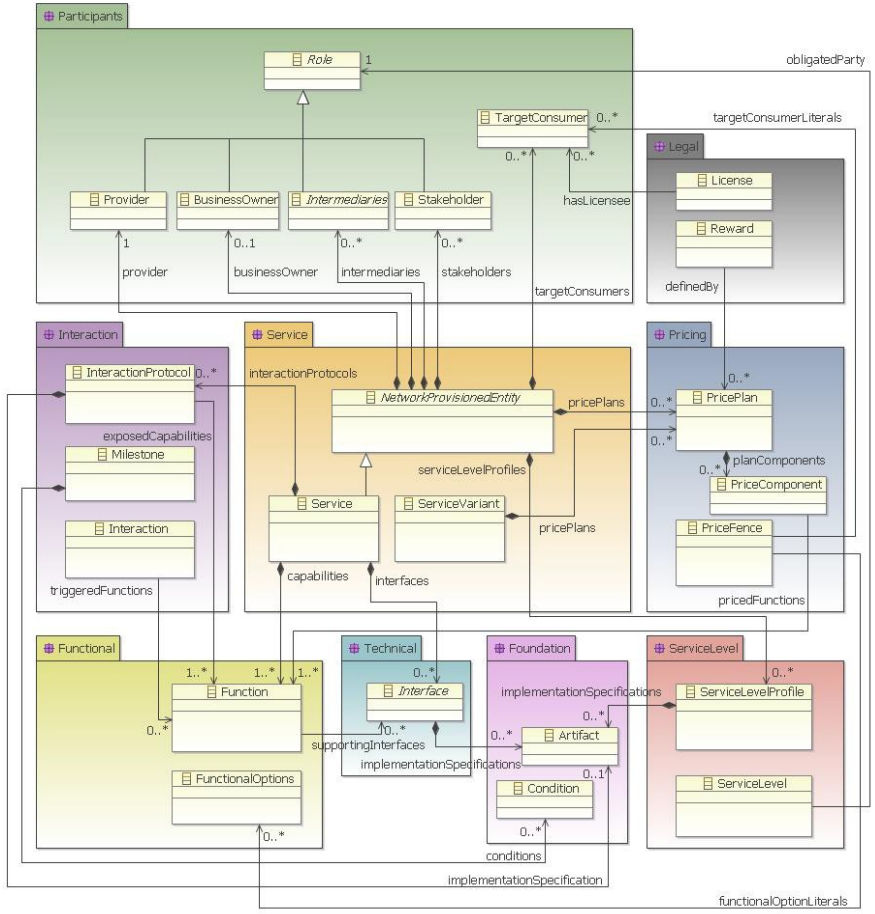




XML serialization

Example: Service description

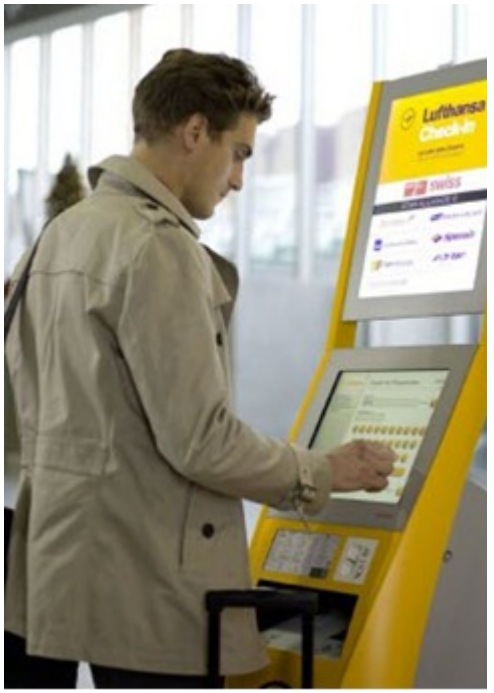
- Enhance service description with business and operational aspects
  - Quality of service, pricing and legal aspects among others
- Transform USDL specification to:
  - UI Views of the MVC
  - Software code in the form of business rules
  - Rules were injected into MVC controllers to specify periods (i.e. days/hours) when ITIL IMS was available to users



XML serialization

USDL

Unified Service Description Language



- Master data model for services
- Describe various types of services
  – professional to electronic services
- Holistic:
  business aspects such as ownership
  and provisioning, pricing and legal aspects,
  in addition to technical aspects

- http://www.genssiz.org/research/service-modeling/alpha-usdl/
- http://www.linked-usdl.org/
- http://www.internet-of-services.com/

Code generation

- The transformation of ISS' models into code
  - Builds upon our previous work
  - Uses XSLT and XPath
- Code organized according to an adapted version of the MVC pattern
  - Called MVC for Services
  - Later deployed in a PaaS

MVC for Services

- Model
  - Code of the data models of the service architecture
  - i.e. ’what’ dimension
- View
  - Instructions of the UI models of the architecture
  - i.e. ’who’ dimension
- Controller
  - Code of the blueprint and USDL models of the service architecture
  - i.e. ’how’, ’why’ and ’when’ dimensions

MVC for Services

- Controller generated from a service blueprint
- Illustrates the control-flow

```
def create
  @incident = Incident.new ( @params [ 'priority' ] )
  @incident.date = Date.today
  if @incident.priority == "1"
    redirect to : action => 'NotifySeniorStaff'
  else
    redirect to : action => 'InitialDiagnosis'
  end
end
```

Service deployment using PaaS

- In our experiment we selected the Heroku platform
  - Ease of use, automation and reliability for Web applications

- The following commands were retrieved and automatically executed to deploy the ITIL IMS:

```
$ heroku create <imservice> –stack cedar
$ git init # Initialize the code repository
$ git commit ... # Several commits were made
$ git push heroku master
```

- Since the PaaS used MVC, it knew exactly where all the necessary directories, models, views and controllers of our ISS were located

Conclusions

- Results
  - SEASIDE systematic methodology for ISS development
- The approach is suitable for
  - The ’massification’ of Internet Self-Services
  - Reduces development complexity and costs
  - Reduces time to market
- Findings
  - Applicability of integrating EA, MDD, MVC, and PaaS to support a step-by-step guidance for ISS development