

e-Framework Service Factorisation

Wilbert Kraan, CETIS

London, 24 January, 2006

Overview

Services' neighbour concepts

- Framework and domain
- Reference model
- Service implementations

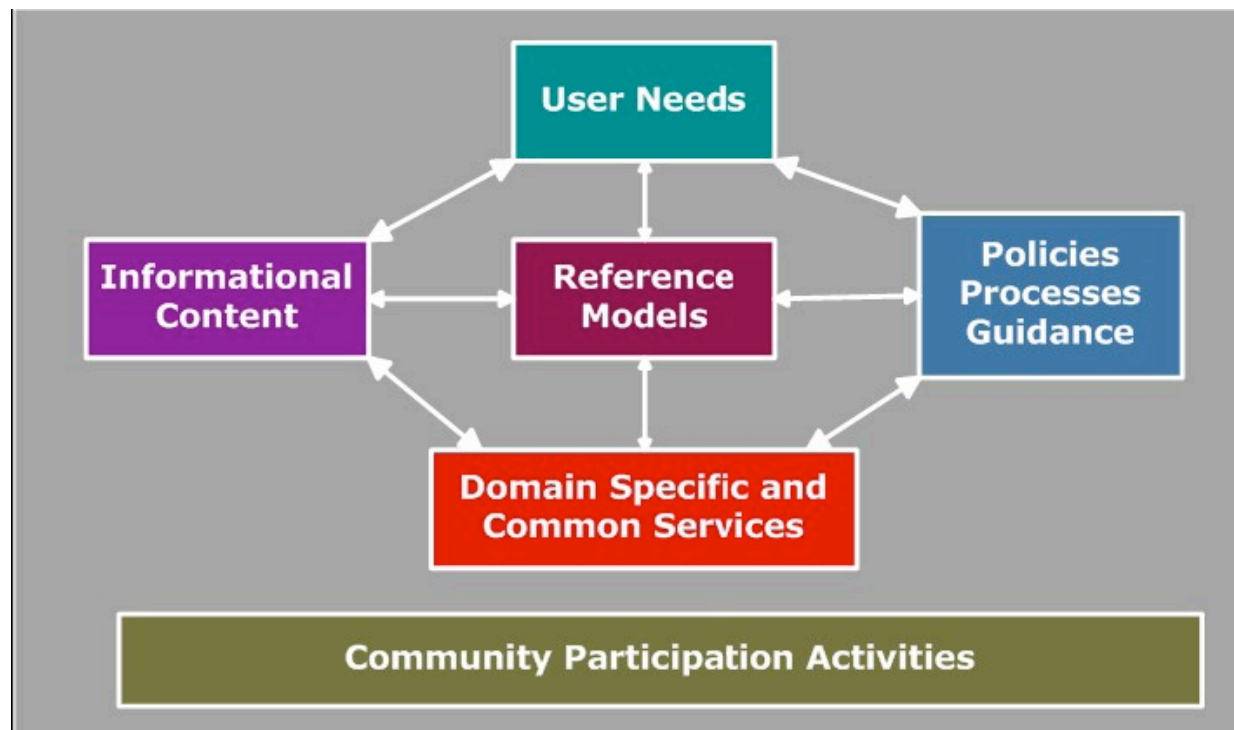
Service

- Definition
- Description example

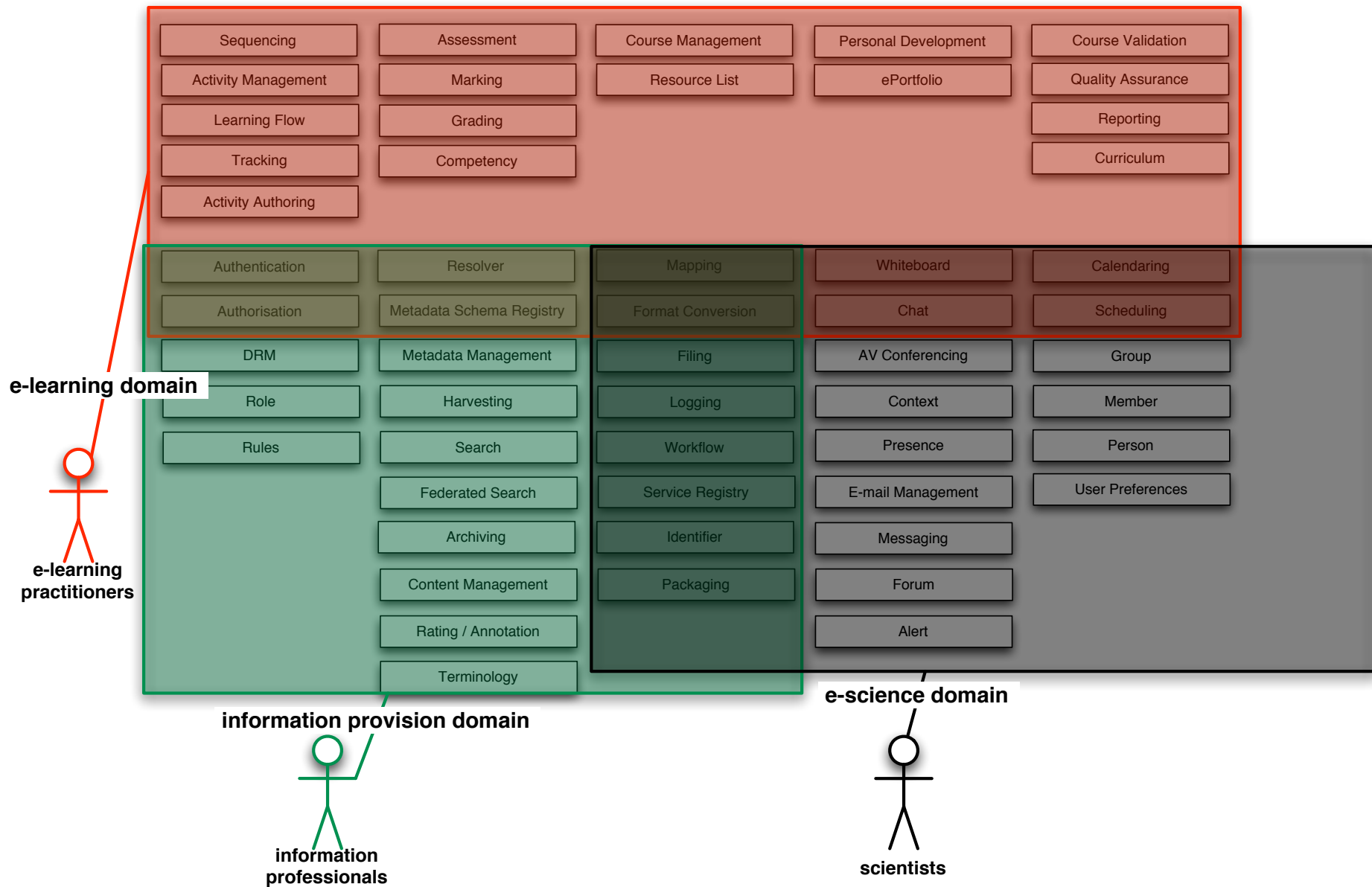
Factorisation hints

Services and neighbours

Framework broad:



Services and neighbours



Reference models

Broad & abstract:

- defined by services that they combine
- aimed at a particular process or workflow
- joins services with requirements
- no limit on the number of reference models in the framework
- can overlap with other reference models

Reference models

For example:

- *Name*: time management
- *Domains*: e-learning, e-science, e-admin
- *Description*:
 - The time management reference model deals with the problem of sharing and co-ordinating the schedules of people and resources in an organisation.
 - It describes a workflow in which various actors can view schedules, edit them, or request an edit in them.
 - The purpose of the model is to make it easier to co-ordinate people and resources such as rooms, equipment and documents

Reference models

time management II:

– *Structure and Organization:*

- Services:

- Scheduling
- Calendaring
- Authorisation
- Messaging (and, through that, Email Management, Chat, etc.)
- Person
- Group
- Alert

- Resources:

- people, rooms, projectors, short loan books, etc.

Reference models

time management III:

– *Functionality:*

- Data

- calendars, authorisation attributes, person & group data, etc.

- Processes

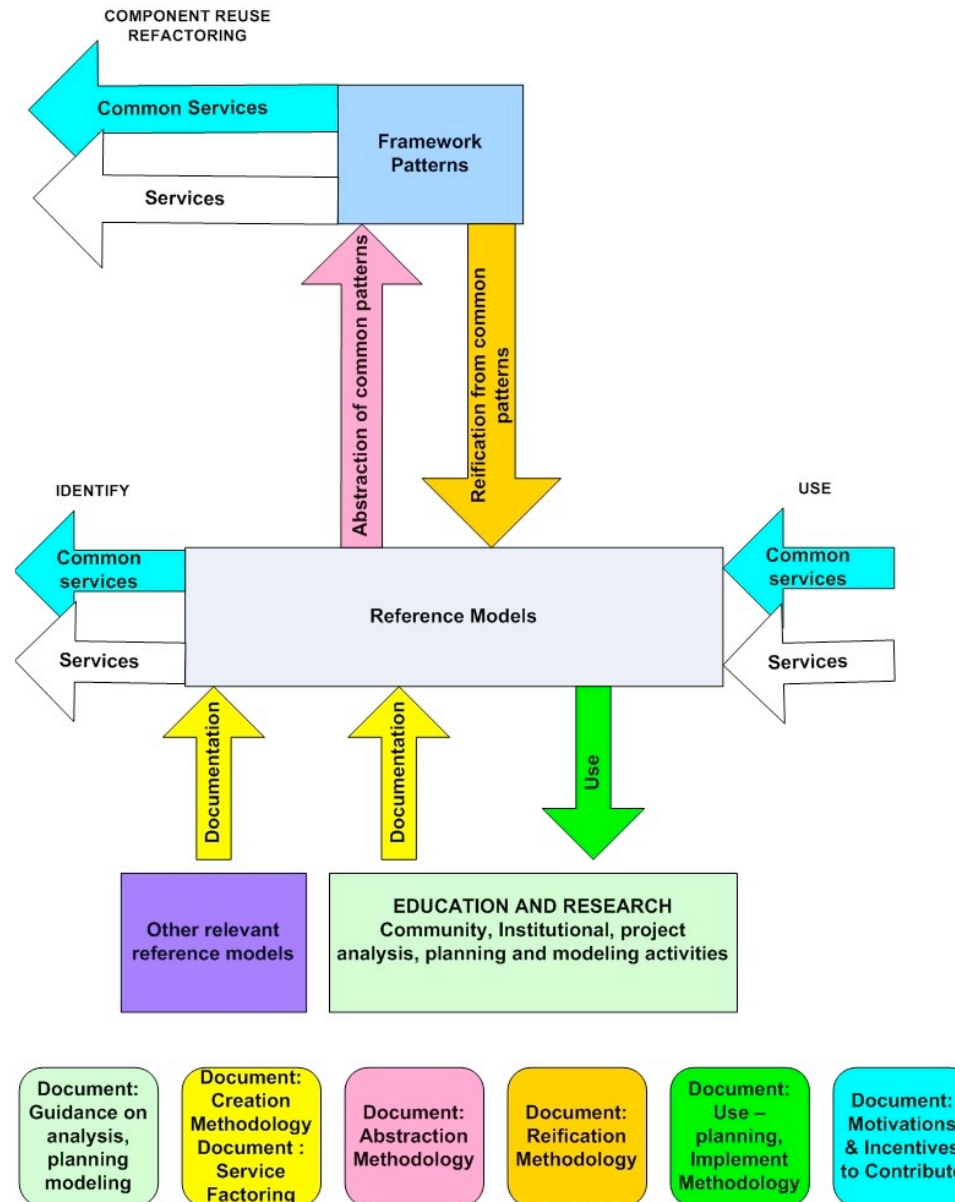
- the orchestration of data, actors and services

- Workflow

- the steps in a process

Use cases, strength analyses and more ...

Why do reference models matter?



Services and their implementations

Service:



Calendar

Agent:

`webdav://cetis.ac.uk/members/core/
wilbert.ics`

a Service:

is a functional resource

is documented by a service description

contains one or more service definitions:

- behaviors and operations
- interfaces

may be used in multiple reference models

may be used in multiple service patterns

may be used in multiple applications

may be related to other services

a Service:

should not overlap in functionality with other services

is implemented via an *agent* through a *design*

- may be many possibly equivalent, independent designs

Service description example

Name

- Calendaring

Description:

- Supports the sharing of calendars, such as personal calendars and course timetables.

Scope & definition:

- A Calendaring service provides access to Calendars, including course timetables. A Calendaring service should support creating, reading, updating and deletion of calendars for persons (staff, students), and groups (courses, modules, departments).

Service description example

Applicable standards:

- IETF RFC 2446 - iCalendar Transport-Independent Interoperability Protocol (iTIP)
- IETF RFC 2445 - Internet Calendaring and Scheduling Core Object Specification (iCalendar)
- IETF RFC 2518 - HTTP Extensions for Distributed Authoring -- WEBDAV

Factorisation

Iterate, iterate, iterate

- it all depends on experience

A Service is not a Component

- a functional part of a system need not be exposed

A Service may be realized in different ways

- webservices aren't the only way

Granularity

- often too coarse; they resemble reference models and overlap with other services

Factorisation

Build on prior work (e.g. look at existing standards)

- don't re-invent wheels

Services Shouldn't Have Dependencies

- you should be able to implement just the one service

More Brains Is Better

- involve all stakeholders

Use Scenarios To Test Proposals

- walk through a process to test assumptions

Thing Big and Think Small

- consider all implementors