

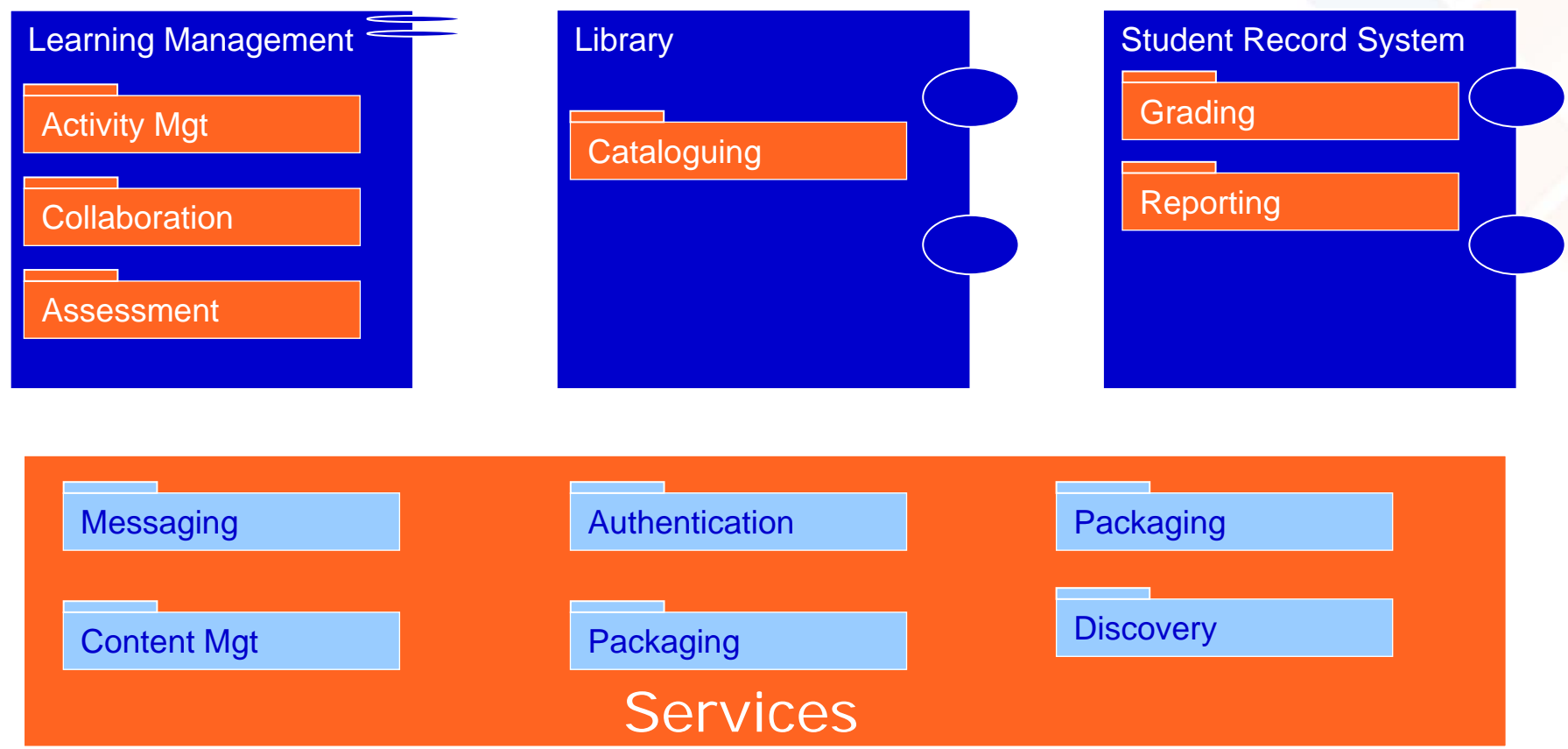


# Service Usage Models

Phil Nicholls ([pjn@psydev.com](mailto:pjn@psydev.com))  
12<sup>th</sup> February 2007  
e-Framework Conference

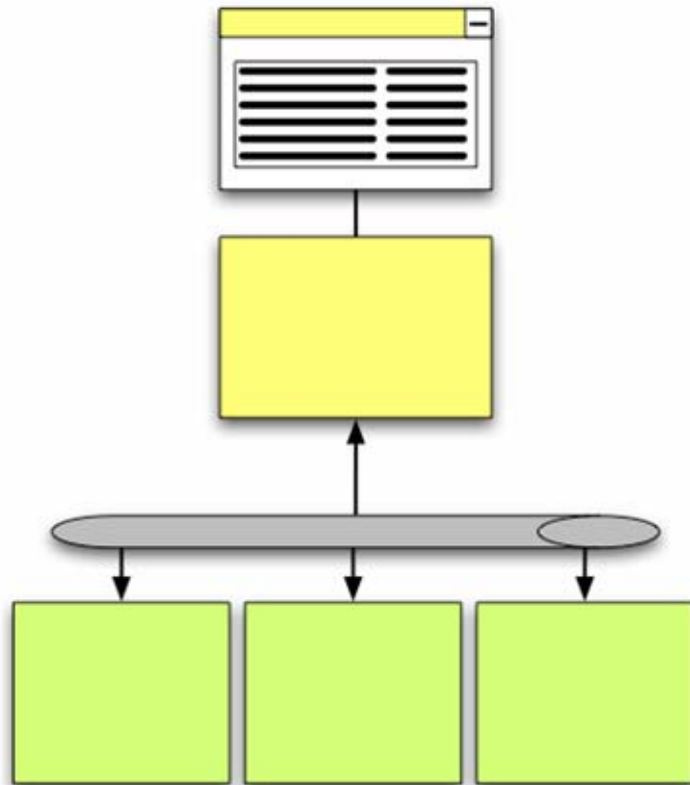
- About SUMs
  - Service Oriented context
- The e-Framework's view of SUMs.
  - CORE sums
  - Purposes of SUMs
  - Submission
- Design of SUMs and difficulties
  - Avoiding the monolithic application
  - Principles of Service Design

# Service oriented approach



Services need well defined interfaces so all components can access them.

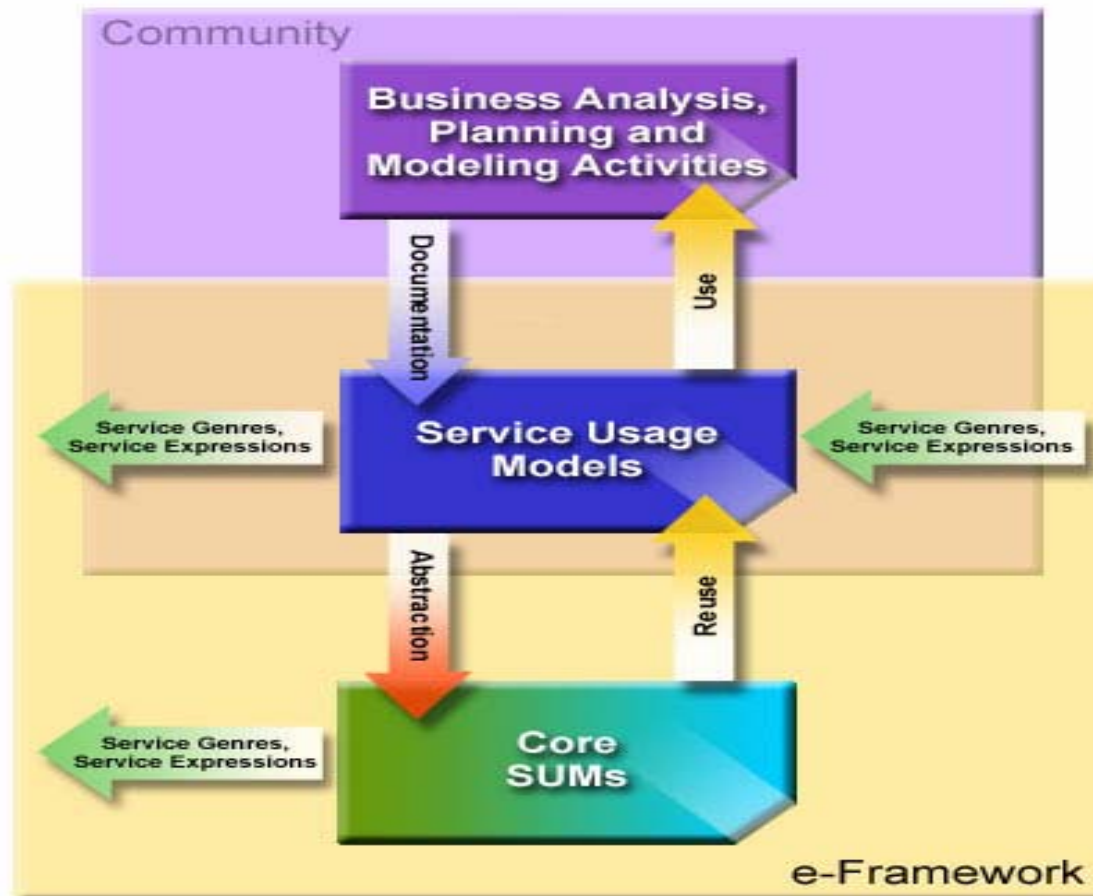
# How do services help build applications?



- Presentation and workflow constructed from multiple shared services
- Data and business function encapsulated in services

- aimed at a particular business process or workflow
- defined by services that they combine
- joins services with requirements
- different levels of granularity, some high level and abstract, some detailed and targeted at implementation
- no limit on the number of SUMs in the framework
- can overlap with other SUMs
- Might commonly occur in different domains

# Where do SUMs come from?



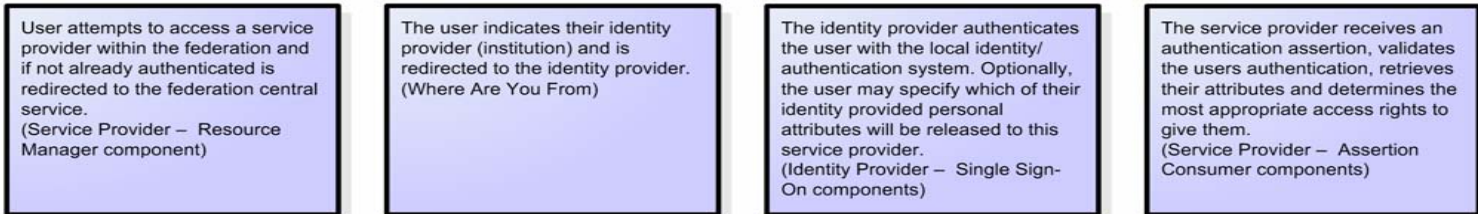
- For example:
  - *Name*: Federated Access Management
  - *Domains*: e-learning, e-science, e-admin, digital library, repositories
  - *Description*:
    - The federated access management SUM deals with managing user identities and access privileges across organizational boundaries and between institutions
    - The SUM describes the set of functionalities and workflows that allow a user within a federation to authenticate with their identity provider before accessing services provided within the federation

# What is a SUM?

## MAMS Federation Authn. Access to Service SUM

Tuesday, January 23, 2007

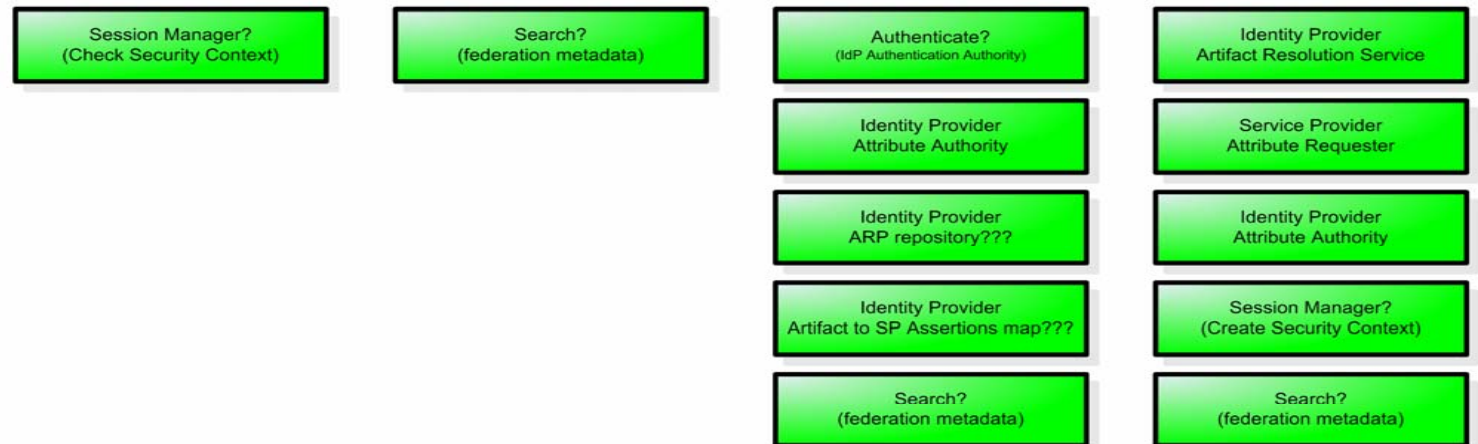
### Business Requirements Summary



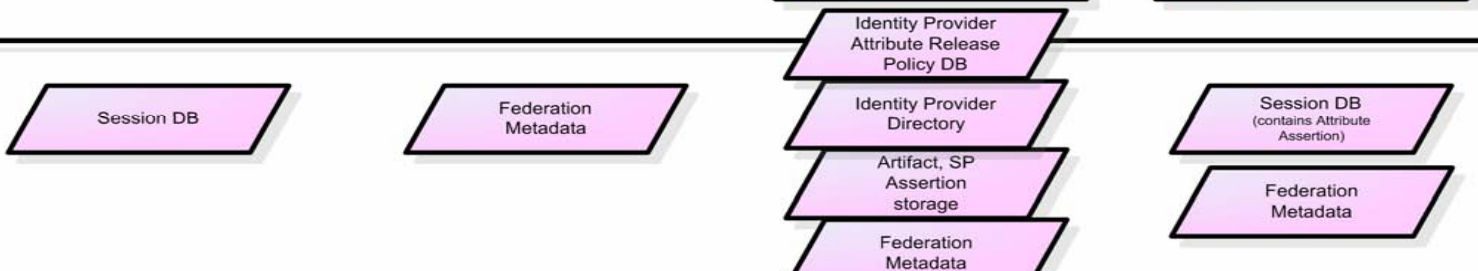
### Business Process Name



### Service Expressions

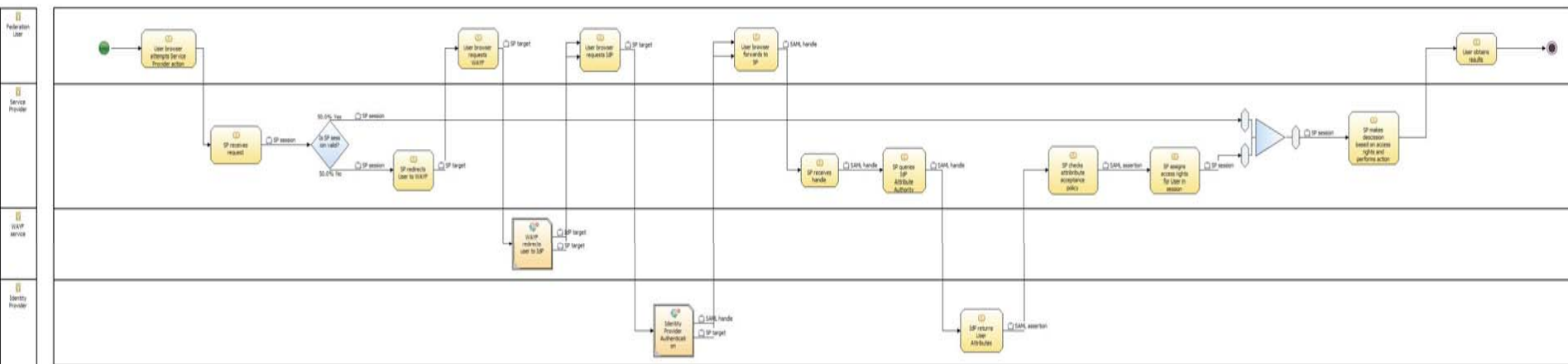
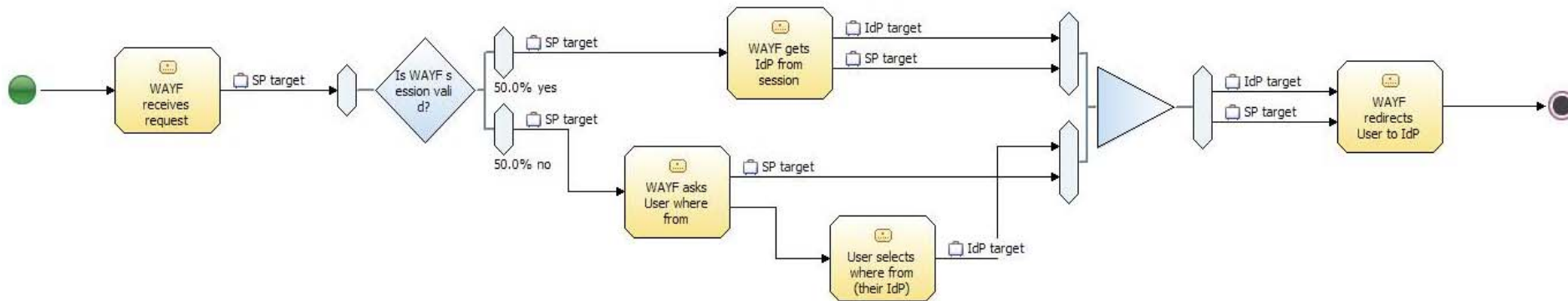


### Data Sources



# What is a SUM?

## Processes and choreography...



# What is a SUM?

- So a SUM is a combination of:
  - Processes
  - Services
  - Data
- A SUM shows:
  - Workflows
  - Choreography
- A SUM links requirements to services

- All SUMs have a purpose:
  - “Exemplar” – High level, abstract used to explain a concept, or provide general guidance
  - “Application” – Implementation plan, detailing combinations of specific expressions; used by a coder
  - “Model” – an exploratory model of business processes and services – somewhere between the two
- Most notable difference is the level of detail

- Exemplar
  - Built from Genres (Abstract)
- Application
  - Built from Expressions (Detailed messages)
- Model
  - Mostly one or the other, with placeholders

- A CORE SUM is:
  - Commonly Recurring
  - Occurs as a part of two or more SUMs
  - Identified through practice
  - Can exist at Genre or Expression level
  - Was once called a Service Pattern
  - Quality Assured by the e-Framework

- Usual template based process
  - Download and complete template
  - Submit via website
  - SUMs are 'registered', so little QA involved
  - Contact editors or colleagues for help
  - <http://www.e-framework.org>

# Designing a SUM

- Assumption: there is some sort of problem to solve:
  - Maybe a business process is only served by a “monolithic” application
  - Uses humans and there is a desire to automate
  - Information duplication
  - Widespread across many institutions
  - “It’s just interesting”
- Solution: Map out and understand the problem space and/or develop some software

# Designing a SUM

- Design a SUM when:
  - Need more than one service
  - Otherwise, design a service...
- If we are already in a services based world:
  - Identify the services
  - Identify the workflows and choreography
- If we are not in a services world:
  - Beware of monoliths!

# Difficulties...

- Which functions come out as services?
- How to define the interfaces to the services?
- Granularity?
- Authentication?

- A service is needed when more than one application needs to use a function
  - Think of business functions rather than software functions
  - Is the function useful to other applications?
  - If a function is only ever used in one specific application, it is not worth pulling out
  - Need to try to do cross application analysis to pull out common functions

- Factoring is easier with experience...
- Some questions that might help:
  - Is there already a standard or specification in the area?
  - Is there already a Genre or Expression in the e-Framework?
  - Does it make sense for the function to be autonomous? Is it useful on its own?

- Boundaries are explicit
  - Clear interface definition (e.g. WSDL)
- Services are autonomous
  - Independently deployed, versioned and managed
- Share contracts, not classes
  - Services separate data from behaviour
  - Changing contracts is expensive
- Compatibility is based upon policy
  - Cannot communicate everything in the interface – this is where the specs / standards come in

# Defining a Service Interface

- Guided by service tenets
  - Well defined message that dictates the action to be taken (think business)
  - Strong types (passed as documents)
  - Use acknowledgements
  - Encapsulation is critical – avoid blurring public and private
  - Avoid RPC interfaces
  - Avoid loose interfaces (e.g. query() and execute())

- Builds on previous points
  - Need to think in terms of business objects and business processes
  - Coarse grained functions
    - Avoid get/set methods
    - Do not call lots of methods to change data state
  - Cohesion according to goal / function
  - Data as documents, not lots of primitives

## Another brief note...

Factoring today is different from factoring tomorrow...

- It is possible to design “for the future”
- Think “re-usable component”
  - If something is perhaps not appropriate for being a service, it might be in the future
  - Putting service interfaces onto existing libraries isn’t necessarily a chore
- Build in a modular way

# What about Authentication?

- Authentication can be a problem as it is ingrained into ‘everything’.
  - Some services require authentication tokens or strings
    - Define in SUM where and how the token is generated
  - Some services refer to external authentication sources / services
    - Include a specific authentication service in the SUM – unless it’s private to a specific service.

# New Genres and Expressions

- In the course of problem solving, new Service Expressions (and maybe Genres) might emerge
  - This is not a bad thing
  - Tell us about them!
- The e-framework is not static; it will evolve over time
  - e-Framework team will track maturity and usage

- About SUMs
  - Service Oriented context
- The e-Framework's view of SUMs.
  - CORE sums
  - Purposes of SUMs
  - Submission
- Design of SUMs and difficulties
  - Avoiding the monolithic application
  - Principles of Service Design

Any Questions...?