

Intelligent Technologies Module: Ontologies and their use in Information Systems

Alex Poulouvassilis

November/December 2009

Overview of this lecture

1. What is an ontology?
2. How are ontologies developed?
3. Reasoning over ontologies
4. Usage Scenarios for Ontologies in Information Systems:
 - Supporting personalisation
 - Ontology as global schema
 - Enabling interoperability between different systems
5. Using Ontologies to support Personalisation

1. What is an ontology?

- An ***ontology*** is a formal representation of a set of concepts, the properties of such concepts, and the relationships between them
- Ontologies can be used to represent knowledge about a particular domain, and provide a common vocabulary for such knowledge to be shared, by humans or by computers
- Historically, ontologies have their origins in Philosophy and in particular the study of “existence”
- This was taken up in the 1980s by researchers in Artificial Intelligence, who were concerned with using domain knowledge to enable automated reasoning
- Tom Gruber’s work in the 1990s brought ontologies into mainstream Computer Science

What is an ontology?

- Ontologies are typically specified in languages that are independent of any particular data modelling language or implementation e.g. UML, RDF/S, OWL
- So they operate at the ***semantic*** level, rather than at the logical or physical level of information systems
- Because of this, they are well suited to
 - supporting systems' adaptation and personalisation, based on knowledge of the user
 - supporting queries to knowledge-based services
 - integrating heterogeneous data
 - enabling interoperability between different systems

What is an ontology?

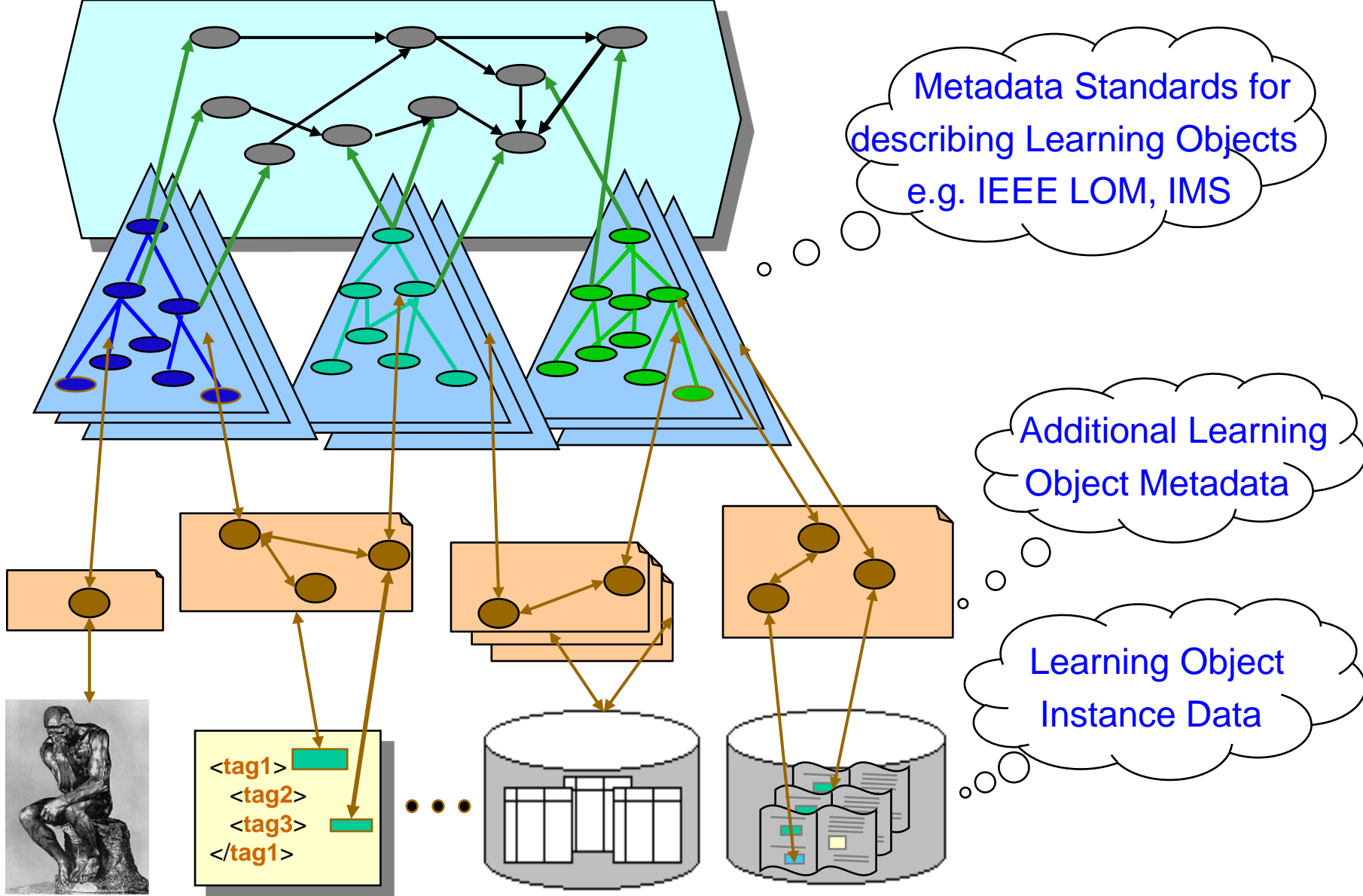
- Ontologies are being applied in fields such as education, bioinformatics, medical informatics, linguistics, geographical information systems, manufacturing, human resources ...
- Michael Zakharyashev's ***Semantic Web*** module discusses in detail the historical, mathematical and practical foundations of defining and reasoning with ontologies

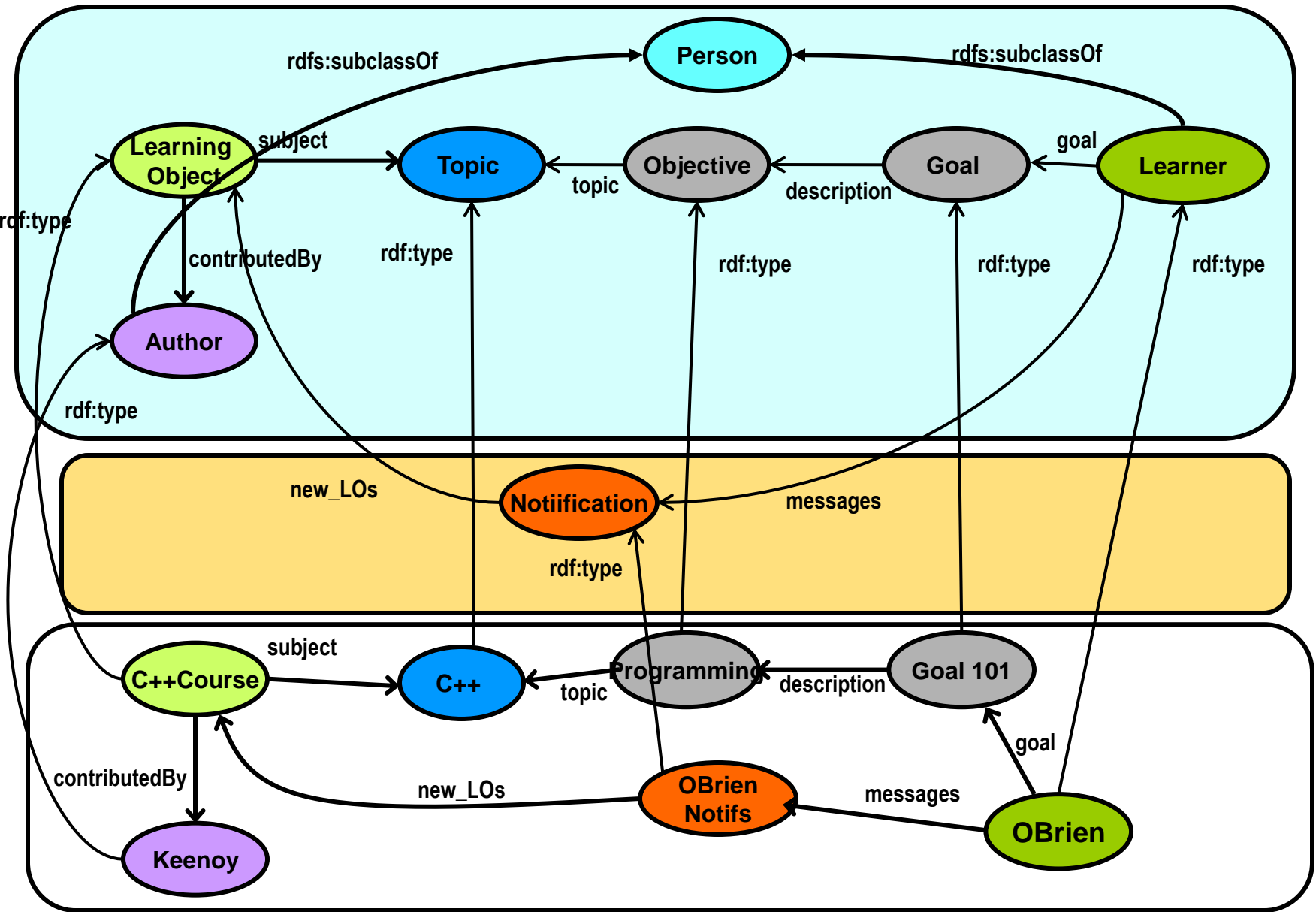
2. How are ontologies developed?

- Developing an ontology requires definition of:
 - the concepts; concepts are also known as classes
 - the subclass/superclass relationships between classes, defined as follows:
 - a class A is a subclass of a class B if every instance of A is also an instance of B.
 - the properties and relationships of classes; these are also known as slots
 - the instances of classes, properties and relationships

2. How are ontologies developed?

- Ontologies typically have three layers:
 - (i) an “upper” layer that may conform to one or more existing metadata schemas or standards;
 - (ii) a “middle” layer, which may be more application-specific; and
 - (iii) an “instance” layer
- For example, in an Education application, we may have the following three layers:





Some example data (in RDF/S)

How are ontologies developed?

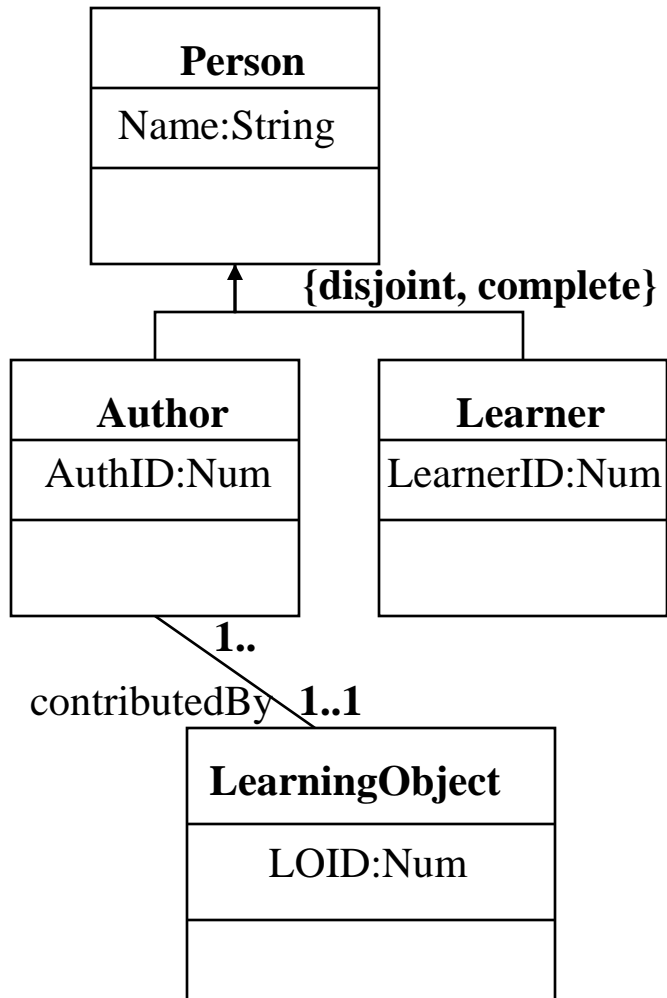
Noy and McGuinness (see Optional Reading section at the end of these notes) describe the following methodology for developing an ontology:

- Step 1. Determine the domain and scope of the ontology:
 - What is the domain that the ontology should cover?
 - What is the ontology going to be used for?
 - What kinds of queries will it need to answer?
 - Who will use it?
- Step 2. Consider extending an existing ontology or ontologies, rather than developing one from scratch
- Step 3. Enumerate important terms that should appear in the ontology

How are ontologies developed?

- Step 4: Define the classes and the class hierarchy (might be done top-down, bottom-up, or a combination of these)
- Step 5: Define the properties and relationships of the classes – the slots
- Step 6: Define the facets of the slots. Slots can have different facets e.g. describing the type of value they can have, the permissible values, and the number of values (cardinality).
 - Common value types are String, Number, Boolean, Enumerated, other classes
- Step 7: Create the individual instances of the classes and define their slot values

Using UML Class Diagrams as an ontology representation language



Mathematical encoding, in terms of sets and relations:

- $\text{Author} \subseteq \text{Person}$
- $\text{Learner} \subseteq \text{Person}$
- $\text{Author} \cap \text{Learner} = \emptyset$
- $\text{Author} \cup \text{Learner} = \text{Person}$
- $\text{domain}(\text{contributedBy}) = \text{LearningObject}$
- $\text{range}(\text{contributedBy}) = \text{Author}$
- $\text{domain}(\text{AuthID}) = \text{Author}$
- $\text{range}(\text{AuthID}) \subseteq \text{Num}$
etc.

Case Study A: Use Cases for Ontologies in Information Fusion, Kokar *et al.* 2005

- Motivation for this work:
 - using ontologies to describe and query heterogeneous sensor data
 - development of use cases for ontologies in this kind of ***information fusion*** setting
- They develop use cases both for the fusion process itself, and for the development of information fusion systems
- Their paper [for you to read] briefly discusses ontologies in Section 2, and particularly their basis for the Semantic Web
- It then briefly discusses the OWL ontology language, and contrasts ontology languages with syntactic languages such as XML and XML Schema
- They then present two fragments of their Situation Awareness (SAW) Ontology, whose aim is to support research in the area of information fusion

Ontologies in Information Fusion

- They use UML to represent these two ontology fragments – although it is not as expressive as the OWL language, for example, UML is easy to understand by humans
- The first fragment models the main classes of the situation awareness domain, namely: Situations, Goals, Objects, Relations, Relation Tuples, Attributes, Attribute Tuples, Event occurrences, Value Functions for attribute and relation tuples, and Events that define these functions
- The second fragment models Event Streams. An event stream consists of a set of Events. An event is concerns a set of Objects. An Object has a set of attributes.
- See page 4 of

<http://www.fusion2004.foi.se/papers/IF04-0415.pdf>

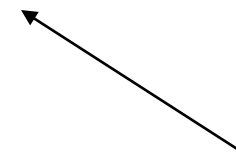
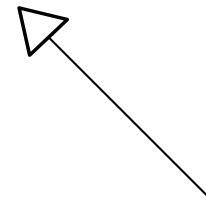
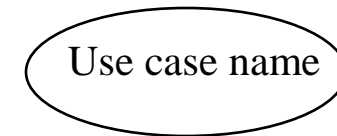
Ontologies in Information Fusion

- Section 3 of their paper presents the concept of a Use Case, using the UML notation for defining use cases
- They then present use cases both for the fusion process itself, as well as for the development of information fusion systems
- See page 5 of

<http://www.fusion2004.foi.se/papers/IF04-0415.pdf>

UML Use Case Diagrams

- Actor: role played by a person, or another system, that interacts with this system
- Use case (how the system is used)
- Arrow: connecting actors with use cases (denoting usage), or use cases with other use cases (denoting generalisation) or actors with other actors (generalisation)
- Arrow for <<include>> and <<extend>>



Case Study B: Developing the SeLeNe User Profile

- Motivation for the SeLeNe project (funded by EU FP5):
 - There are a huge number of learning resources available on the Web
 - There are diverse communities of learners, geographically distributed and with a range of educational backgrounds and learning needs
 - Tools are needed to allow the discovery, sharing and collaborative creation of learning resources
 - Semantic metadata describing these resources can enable advanced services more powerful than traditional Web services

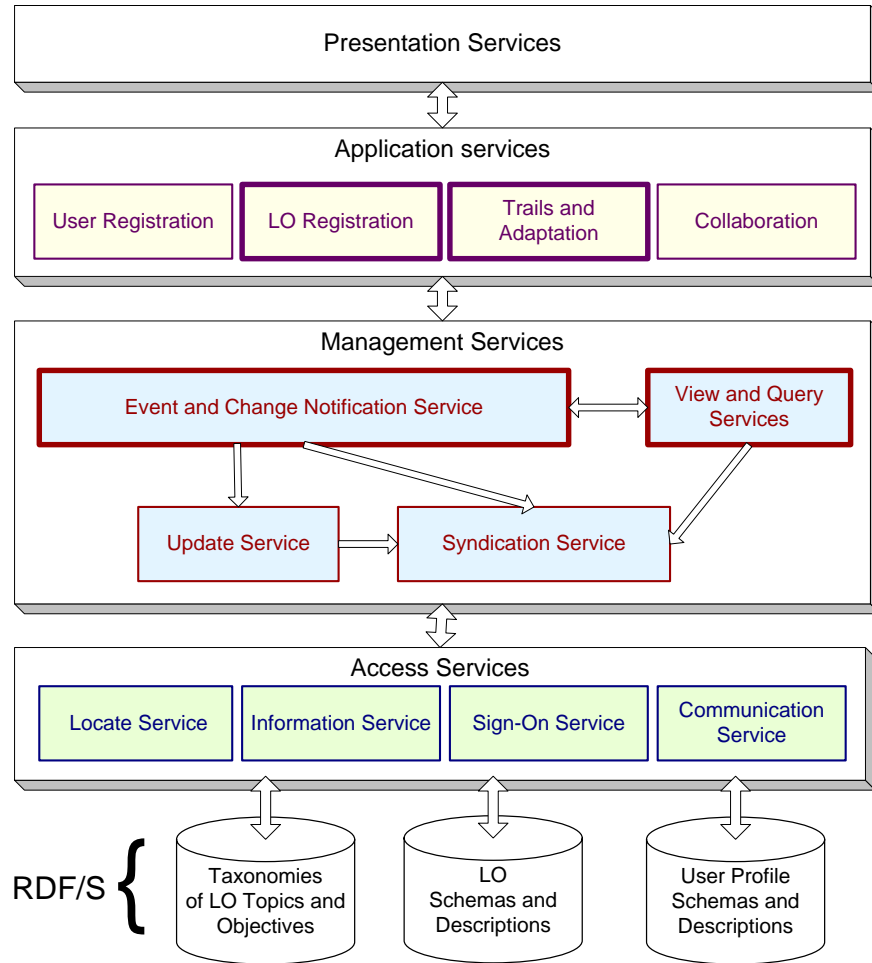
What is a Self e-Learning Network (SeLeNe)?

- A SeLeNe is formed by members of a *learning community*, comprising:
 - instructors, learners and authors*
- The community creates a collection of shared *Learning Objects* (LOs) and *descriptions* of these LOs
- Users register a new LO with the SeLeNe by providing a description of it; some parts of this metadata may be automatically generated
- LOs are described using terms from e-learning metadata standards such as IEEE LOM and IMS
- The LO descriptions are stored in a, possibly distributed, metadata repository

What is a Self e-Learning Network (SeLeNe)?

- The SeLeNe system manages the LO descriptions, not the LOs themselves, which may be stored in separate Learning Management Systems
- Generally, learners may vary in their learning preferences, learning goals and prior knowledge
- Other information managed by the SeLeNe system therefore includes learners' personal data, learners' profiles, and information about how LOs can be sequenced into *trails* of LOs

The SeLeNe Architecture



- The system has a layered ***service-oriented architecture*** which provides the full overall functionality of the system
- There are various service deployment options, the most general of which allows services to be distributed across sites in a ***peer-to-peer architecture***
- The project focussed on developing services for LO Registration, Trails & Adaptation, View & Query Services, and Event & Change Notification

Developing the SeLeNe User Profile

- The project began by defining the users' requirements and functionality that needed to be supported by the system [discussed in Section 3 of SeLeNe Deliverable 2.2]
- We then identified the types of descriptions that the SeLeNe metadata needed to support, with respect to both Learning Objects and Learners
- We next undertook an investigation of existing educational metadata schemas, in order to find out to what extent they each covered the SeLeNe requirements
- For the ***SeLeNe User Profile*** we discovered that some elements from existing user profile metadata schemas could be reused; in particular, information from:
 - the IEEE LTSC Personal and Private Information (PAPI) standard
 - the IMS Learner Information Package (LIP)
 - the IMS Reusable Competency Definitions (RCD)

Developing the SeLeNe User Profile

- We therefore included in the SeLeNe User Profile the appropriate elements from the above existing metadata schemas
- And we extended these with our own SeLeNe-specific elements where existing metadata specifications failed to be expressive enough to capture the SeLeNe requirements
- The SeLeNe User Profile is composed of two parts:
 - (i) information explicitly supplied by the user
 - (ii) information collected transparently by the system according to the user's behaviour in interacting with the system

Developing the SeLeNe User Profile

- For (i), the following elements are drawn from PAPI:
 - Learner's Personal Information (e.g. id, name, address)
 - Learner's Preference Information (e.g. accessibility requirements, preferred LO providers) – **extended with additional modelling elements to capture preferred Learning Styles**

and the following elements from IMS-LIP and IMS-RCD:

- QCL – Qualifications, Certifications and Licences received, and details of each (e.g. awarding organisation, level, title)
- Interest, including type of interest (Recreational|Vocational|Domestic) and description
- Competency – **extended with our own modelling elements for capturing users' competencies**
- Goal, including type of goal (Work|Education|Personal), description, priority – **extended with our own modelling elements for users' Learning Objectives**

Developing the SeLeNe User Profile

- For expressing a user's **Learning Objectives** in the case of Educational goals, we used the same Learning Topics taxonomy and Learning Objectives taxonomy as we used for describing Learning Objects

Note: a taxonomy is a simple form of ontology consisting of concepts, subconcepts and terms – no properties or other relationships

- If learners express their learning needs using the same terms as LO providers use to describe their LOs, this enables easier ***matching*** of users' profiles and LO descriptions in order to support personalised search for relevant LOs (compared to using different ontologies)

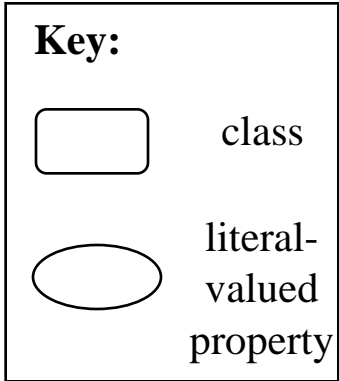
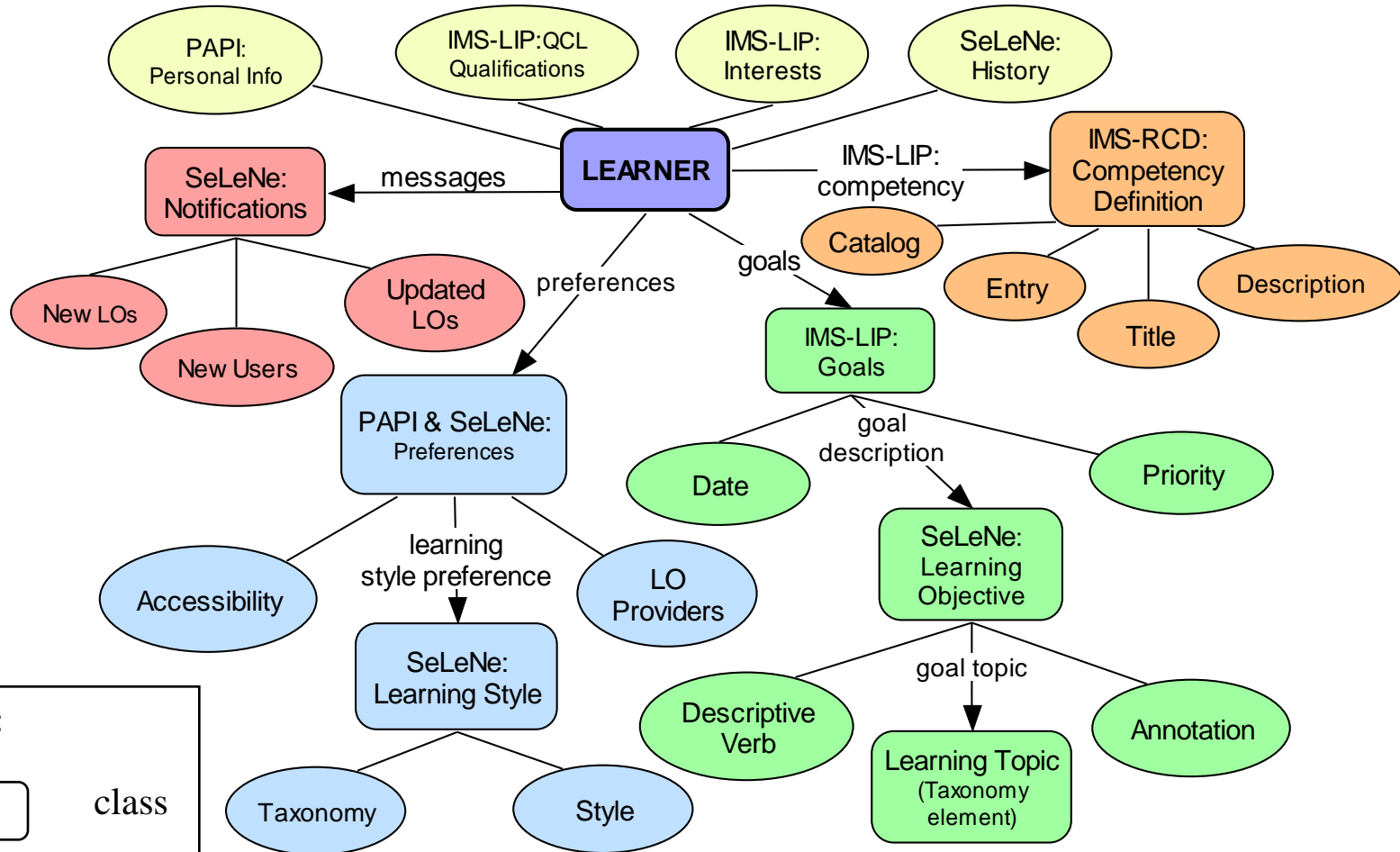
Developing the SeLeNe User Profile

- For (ii) – information collected transparently by the system about the user – this part of the SeLeNe user profile represents the adaptive aspect of the profile
- In particular, a **History** of the user's searches and accesses of LOs is maintained by the system
- this history provides information that can be mined in order to discover information that is useful for personalisation e.g.
 - preferred authors, preferred LO learning styles, LO topics of interest, preferred languages
- thus some of the initially user-supplied parts of the user profile can be automatically updated to represent the user more accurately over time
- automatic updates of a user's profile based on the evolving user's history can be implemented using *Event-Condition-Action rules* associated with each user's profile (see later)

Developing the SeLeNe User Profile

- Also maintained within the adaptive part of the user profile is a **messages** property of the Learner class, which has a class **Notifications** as its range
- Notifications has three properties, representing the three types of event that a SeLeNe user can choose to be notified of:
 - addition of **new LOs** to the SeLeNe repository that may be of interest to the user
 - **update of LOs** that the user has selected to be notified of and
 - registration of **new users** who may be of interest to this user e.g. because they share similar interests or goals
- Again, automatic update of this information in the user's profile can be implemented by means of Event-Condition-Action rules associated with each user's profile (see later)

The Final SeLeNe User Profile



3. Reasoning over ontologies

- This is discussed in detail in the Semantic Web module
- In brief, we can use the underlying mathematical foundations of ontology languages (such as OWL) in order to perform reasoning tasks such as:
 - Classification of individuals e.g. $\text{isa}(\text{Obrien}, \text{Learner}) ?$
 $\text{isa}(\text{Obrien}, \text{Person}) ?$
 - Subsumption of concepts e.g. $\text{Author} \subseteq \text{Person} ?$
 - Equivalence of concepts e.g. $\text{Author} = \text{Person} ?$
 - Disjointness of concepts e.g. $(\text{Author} \cap \text{Learner}) ?$
 - Satisfiability of concepts e.g.
 $\text{domain}(\text{contributedBy}) \cap \text{Learner} ?$
 $\text{range}(\text{contributedBy}) \cap \text{Person} ?$
 - Inconsistency inference e.g. $\text{isa}(\text{Keenoy}, \text{Learner}) \wedge$
 $\text{isa}(\text{Keenoy}, \text{Author}) \wedge (\text{Author} \cap \text{Learner} = \emptyset) ?$

4. Some Usage Scenarios for Ontologies in Information Systems

- Supporting personalisation
- Ontology as global schema, for example:
 - integrating heterogeneous databases under an ontology
 - querying heterogeneous databases through an ontology
 - integrating web search engines using ontologies
- Enabling interoperability between different systems

5. Using Ontologies to support Personalisation

- We will look at two case studies:
 - Personalisation in the SeLeNe system
 - Personalisation in the L4A// system

Case Study C: Personalisation in the SeLeNe system

- There are many LOs available to users of a SeLeNe
- Some LOs will be useful for them while others will not
- SeLeNe envisages several kinds of personalised access to LOs for learners, in order to aid them in sharing and finding useful LOs. We will look at two of these functionalities in this lecture:

Personalised Queries: This functionality aims to present learners with LOs that are relevant to their needs and preferences

Personalised Notifications: This functionality aims to notify learners of additions and updates to LOs, and of registration of new users, that may be relevant to them

(i) Personalised Querying

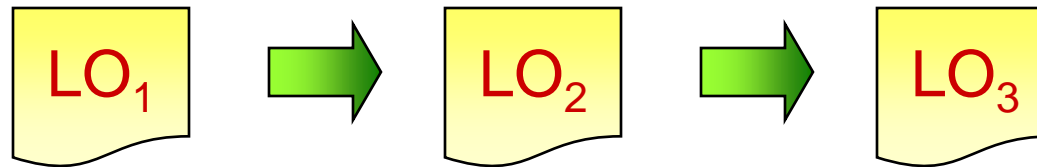
- Personalised querying is made possible by the SeLeNe User Profile, which we discussed earlier
- The LO descriptions are queried using the RQL query language (a precursor of the SPARQL query language)
- Queries expressed in RQL are generated from keyword-based queries that are submitted by users
- The generation of RQL queries takes into account the profile of the user who is submitting the query
 - e.g. if the profile records that the user only speaks English, then the condition `lang:en` is added to the query and only LOs written in English will be returned
 - this is the query ***Filtering*** stage, which takes place before the query is evaluated

Personalised Querying

- The set of LOs returned after query evaluation are then ordered according to their “relevance” to the user – this is the results ***Ranking*** stage
- The information contained in the user’s profile allows factors such as the following to be taken into consideration when determining the relevance of a particular LO to the user:
 - similarity of the LO description to the original query terms
 - whether the user has the prerequisite knowledge for the LO
 - how well the learning objectives of the LO match the user’s learning goals
 - whether the user’s preferred learning style(s) are catered for by the LO

Personalised Querying

- Query results may be presented as a set of *trails* of LOs – trails are suggested sequences of users' interaction with LOs:



- We have defined an RDF representation of trails as a subclass of the RDF *Sequence*, with additional properties `name` and `annotation`
- Such trails can be recommended by SeLeNe users who have the role of Instructors
- Trails of LOs can also be automatically generated by the system based on LOs' descriptions; in particular, based on the `LOM:Relation` field that encodes semantic relationships between LOs e.g. "is a prerequisite of", "extends", "elaborates upon" etc.

(ii) Personalised Notifications

- The SeLeNe higher-level Presentation and Application services (see Architecture diagram earlier) may generate *Event-Condition-Action* (ECA) rules of the form

on event if condition do action

- These are expressed in ***RDFTL*** – an ECA rule language for RDF/S data that we designed as part of the SeLeNe project
- RDFTL ECA rules operate over SeLeNe’s distributed RDF/S repository, much like traditional database triggers over a database
- Such ECA rules enable notification of
 - the registration of new LOs or new users of potential interest to a user
 - changes to descriptions of particular LOs that a user has chosen to be notified of
- ECA rules also allow propagation of changes in a user’s history to updates to the user’s personal profile (as discussed earlier)

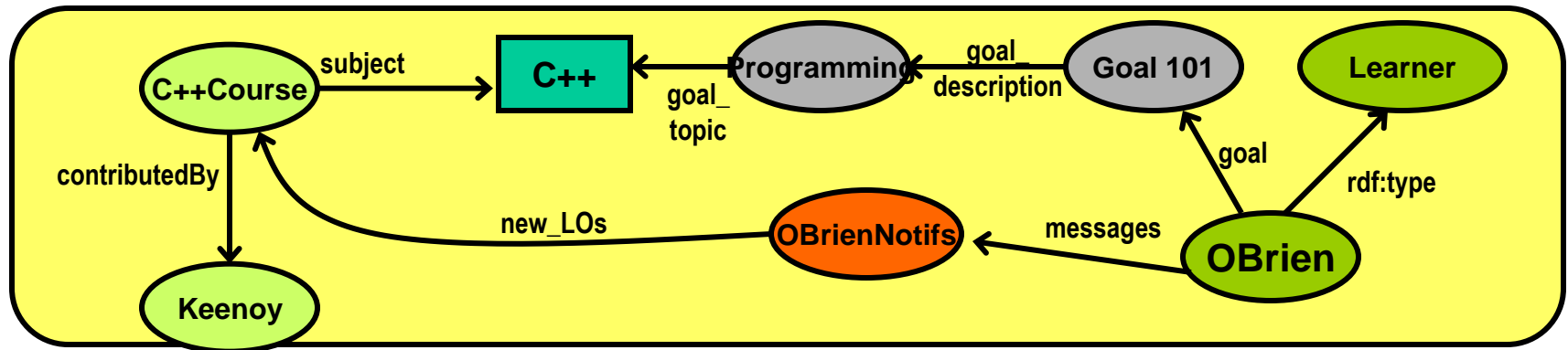
Example illustrating the RDFTL language

```
ON INSERT resource() AS INSTANCE OF LearningObject
IF $delta/target(LOM:subject)
  = resource(http://www.dcs.bbk.ac.uk/users/OBrien)
  /target(ims-lip:goal)
  /target(ims-lip:goal_description)
  /target(selene:goal_topic)
DO LET $notifs :=
  resource(http://www.dcs.bbk.ac.uk/users/OBrien)
  /target(selene:messages)/
IN INSERT ($notifs,new_LOs,$delta);;
```

→ Checks for triggering **event** ✓

→ Checks to see if the **condition** is met (i.e. if the inserted resource is on the topic of one of user OBrien learning goals) ✓

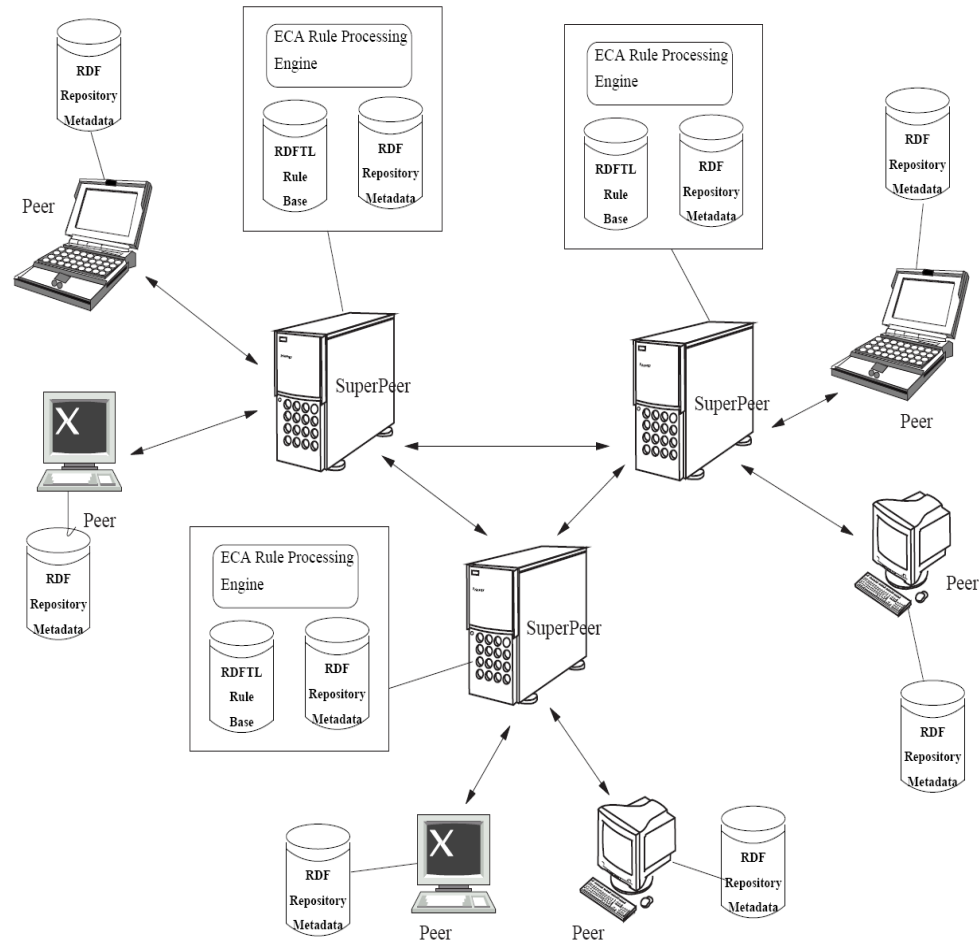
→ Carries out the **action** associated with this rule (i.e. notifies learner OBrien that there is a new LO of interest to him/her) ✓



RDFTL Deployment

- For implementing RDFTL, we assumed a ***peer-to-peer architecture*** comprising a set of peer and superpeer servers
- Each superpeer (SP) server in the network coordinates a group of peer servers
- Peers communicate only with their SP
- SPs communicate with their peergroup, and with all other SPs
- Each peer or SP hosts a fragment of the overall RDF/S data
- Each SP's RDFS schema is a superset of its peergroup's individual RDFS schemas
- The RDF/S data stored at a peer may change over time, and it notifies its supervising SP of changes, so that the SP can update its schema and its indexes accordingly
- At each SP there is an installation of an ***RDFTL ECA Engine***

RDFTL Deployment



RDFTL Implementation

- Each RDFTL ECA Engine at a superpeer provides an “active” wrapper over the set of “passive” RDF/S repositories at its peers, exploiting the query, storage and update functionality supported by these repositories
- An ECA Engine consists of an ***Event Detector, Condition Evaluator*** and ***Action Scheduler***
- A new ECA rule r generated at some superpeer, or at one of the peers in its peergroup, will be stored in a ***Rule Base*** at that superpeer
- Rule r will also be sent to all other superpeers, and will be replicated at those superpeers where events can occur within the peergroup’s repositories that may match the event part of the rule (this can be determined by comparing the event part of the rule with the superpeer’s RDFS schema)

Rule Execution

- Each peer and superpeer manages its own local transaction execution schedule
- Whenever an update u is executed at a peer P , peer P notifies its supervising superpeer SP that u has occurred
- SP determines whether u may trigger any ECA rule in its rule base whose event part is annotated with P 's ID
- If a rule r may have been triggered by update u then SP will send r 's event query to P to evaluate
- If r has been triggered (i.e. the event query evaluates to a non-empty result at P), the condition part of r is next evaluated – generating a set of instances of this condition part for each value of the $\$delta$ variable, if $\$delta$ appears in the condition part
 - values of the $\$delta$ variable arise from the result of the event query

Rule Execution

- If a condition instance evaluates to true, SP will send a corresponding instance of r 's action part (i.e. for the same value of δ) to its peergroup to execute on their repositories
- The instances of r 's action part will also be sent by SP to all other superpeers
- Each superpeer will consult its RDFS schema to determine if the updates it has received are relevant to its peergroup's data, and if so it will schedule and execute these updates
- In summary:
 - execution of an update at some peer may cause events to occur;
 - these events may cause rules that are defined locally to fire;
 - resulting in new updates that are executed locally,
 - and are also propagated to all other sites of the network as well.

Case study D: Personalisation in the L4A// system

- Motivation:
 - Face-to-face guidance and support for career and educational choices has been found to be patchy and uneven
 - Transitions from one stage of education to the next are critical decision points in a learner's journey and better targeted information can make these transitions more successful
 - In this direction, the **L4A//system** aims to support lifelong learners in exploring learning opportunities and in planning and reflecting on their learning
 - It allows learners to create and maintain a chronological record of learning, work and personal episodes – their *timeline*:

Distinctive features

- L4A// is distinctive in that the timeline provides a record of *lifelong learning*, rather than learning at just one stage or period
- Learners can share their timelines with other learners, with the aim of fostering collaborative formulation of future learning goals and aspirations
- The L4A// pilot system was co-designed with FE and HE students, educators and other stakeholders in the London region:
 - A consultation process was undertaken at the start of the project to identify learners' educational goals and career objectives, with the aim of accommodating the needs of different user groups
 - The same groups of learners participated in several phases of formative, and finally summative, evaluation of the pilot

Design & Implementation

- The L4A// user interface provides screens for the user to enter their personal details, to create and maintain a timeline of their learning, work and personal episodes, to search over the course information, and to search over the timelines of other users
- The system architecture consists of two main components:
 - the web portal
 - the web services and related core components
- The web portal consists of several JSP pages, providing users with an interface for the core functions of the system:
 - user and timeline management (creation, modification, etc.)
 - timeline search and course search (within in the L4A// database or through the LearnDirect service)
 - GUI management (registration forms, timeline visualisation etc.).

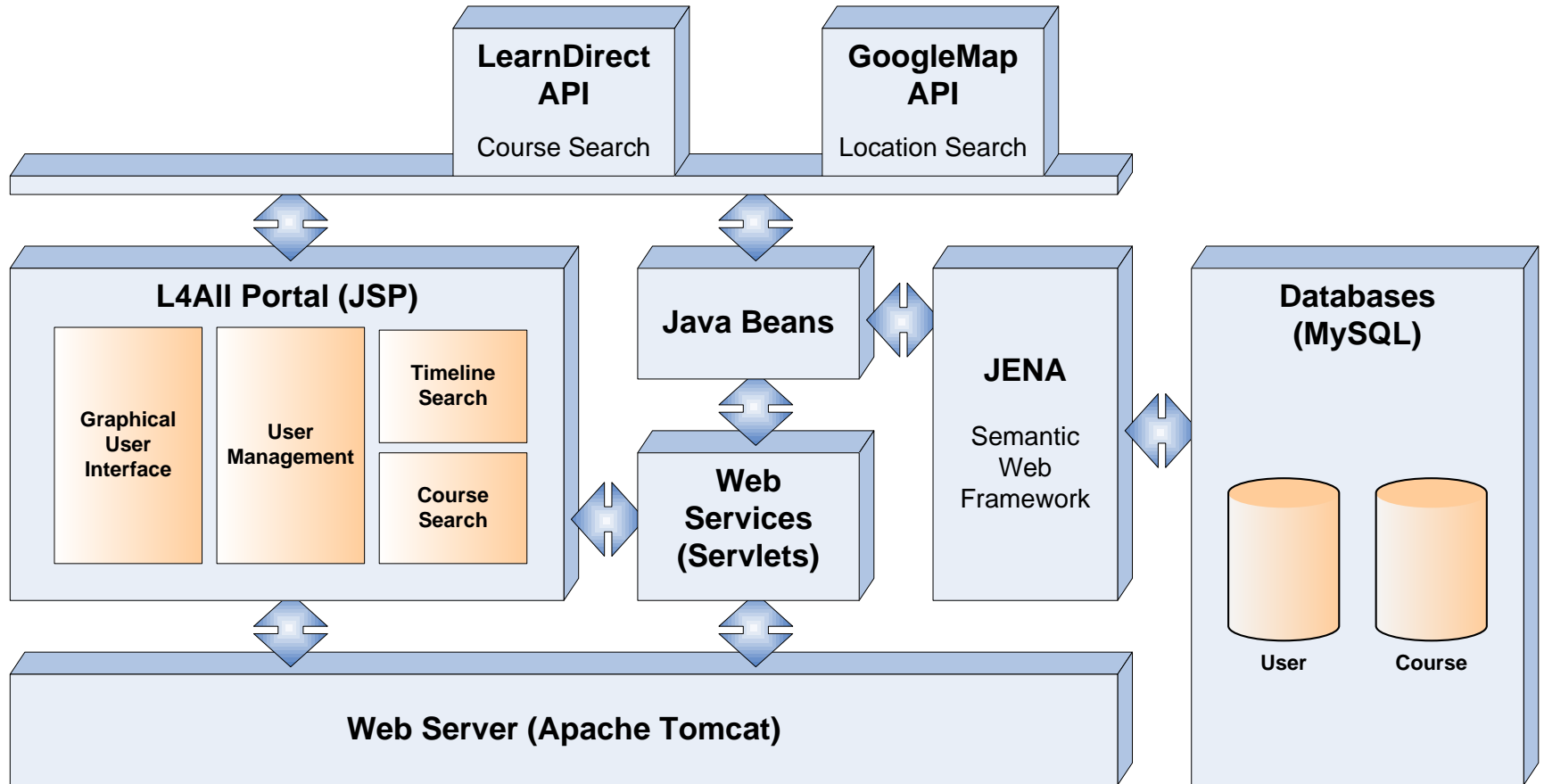
Design & Implementation

- The back-end of the system consists of a set of dedicated web services, implemented as Java Servlets over Apache Tomcat
- These services accept GET and POST requests from the UI components and submit requests to the appropriate Java Bean for processing. The results of the processing are passed back to the calling UI component
- The Java Beans are a set of Java classes providing communication with the databases and integration with external services
- The course and user descriptions are stored in a MySQL database, using the Jena framework which provides mechanisms for storing, retrieving and querying RDF/S data
 - Course and User descriptions are encoded in RDF/S, using elements from the Dublin Core and IMS standards, extended with additional modelling elements for describing L4A//learners and their timelines

Design & Implementation

- The system accesses two external web services:
 - The **LearnDirect API** allows search within their directory of courses. This API is used to give users access to course descriptions, enabling them to add relevant courses into their timeline as new course episodes
 - The **GoogleMap API** provides several services for manipulating location-based information, as well as managing their interactive map. It is used to display locations associated with users (e.g. their current or past location), timelines (e.g. location of particular episodes) and courses (e.g. location of course providers), and for computing distances between locations.

L4A// System Architecture



Evaluation I

- Evaluation of the first L4A// pilot was undertaken with groups of learners from FE and HE
- Learners reported that they enjoyed using the system, that it helped them take a holistic view of their educational and career prospects, and that it had the potential to provide useful support for making career decisions and educational choices
- However, the evaluation also highlighted several areas where further work was required, including provision of more ***personalised support:***

Evaluation I

- In particular, the pilot system supported a search facility over the timelines in the L4A// database
- However, this was based on the occurrence of specified keywords in one or more episodes of the timeline and did not exploit the *sequence* of episodes in the timeline
- Also, the search results were *not personalised* according to the user who was performing the search
- We therefore decided to develop a new timeline search facility that could overcome these limitations
- Our first step was to extend the L4A// ontology so as to associate additional metadata with episodes
- This would allow us to then design a more semantics-based timeline search facility

Ontology Extension

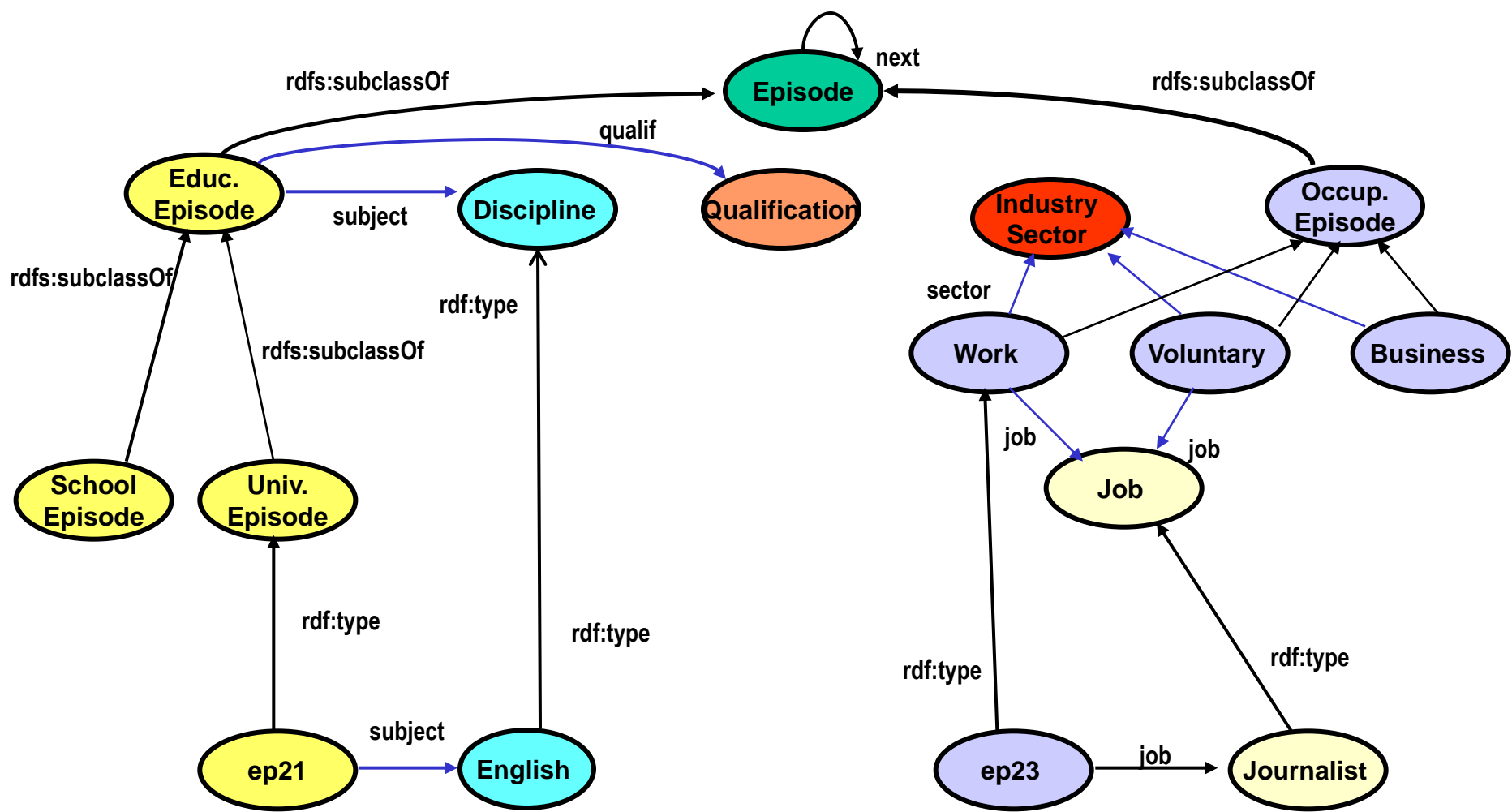
- There are some 20 categories of episode supported by L4A//, which fall into one of three types:
 - **Educational** e.g. attended school, college or university; attended a course
 - **Occupational** e.g. was employed, set up a business, retired, did voluntary work
 - **Personal** e.g. moved, travel abroad, illness, marriage
- We undertook research into existing metadata standards for the first two of these types of episodes:
 - given the aims of the L4A// system, Educational and Occupational episodes are more likely to be the focus of a user's search, rather than Personal episodes

Ontology Extension

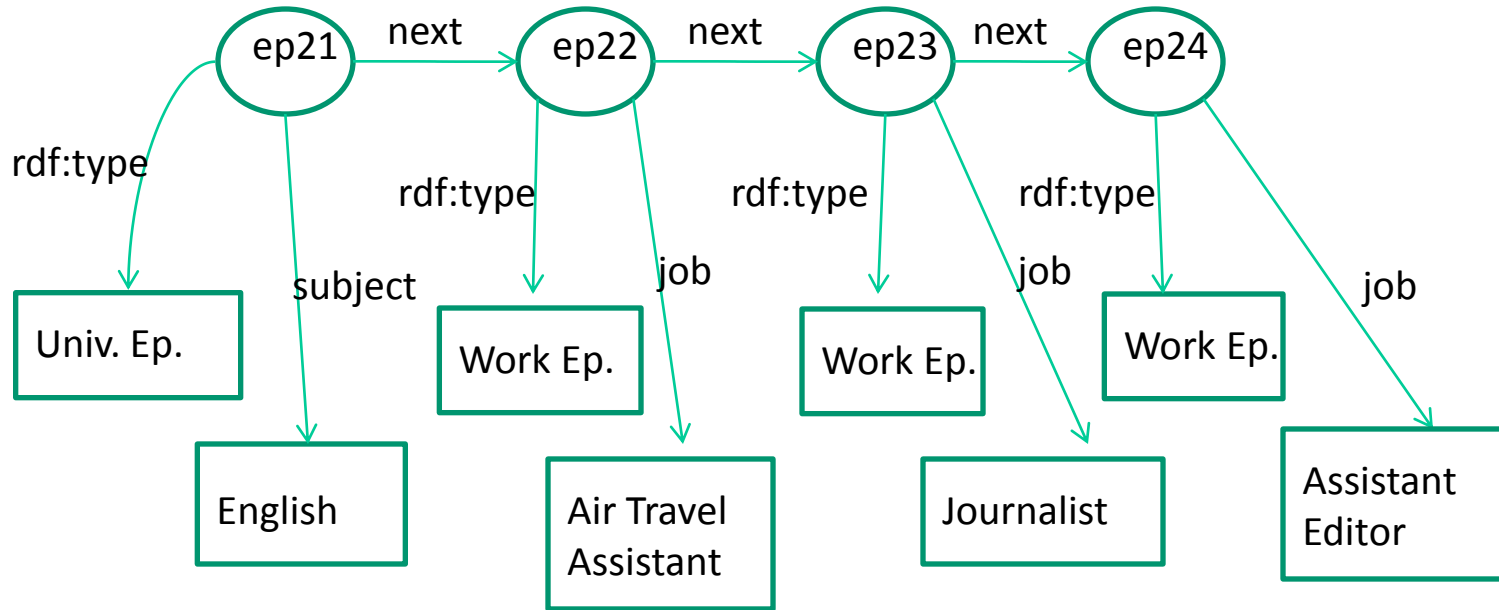
- Following this research, we identified four existing U.K. classifications that were appropriate for associating metadata with some categories of Educational and Occupational episodes:
 - the *National Qualification Framework* (NQF)
 - the Labour Force Survey's *Subject of Degree* (SBJ)
 - the *Standard Occupational Classification* (SOC)
 - the *Standard Industrial Classification* (SIC)
- We expanded the existing L4A// ontology with the concepts and relationships from these four taxonomies, except that we limited their depth to just four levels
- We extended the functionality for creating timeline Episodes so as to allow the user to select a primary and possibly a secondary classification for certain categories of episode. In particular:

Ontology Extension

- **Qualification** is used as a primary classification for all educational episodes: school, college, university, course and degree. Qualifications are drawn from the NQF taxonomy
- **Discipline** is used as a secondary classification for all educational episodes. Disciplines are drawn from the SBJ taxonomy:
 - Episodes are assumed to refer to one discipline only. Support for multi-disciplinary or cross-domain episodes has not yet been considered
- **Industry sector** is used as a primary classification for some of the categories of occupational episodes: work, voluntary and business. Industry sectors are drawn from the SIC taxonomy
- **Job** is used as a secondary classification for some categories of occupational episodes: work and voluntary. Jobs are drawn from the SOC taxonomy



Fragment of RDF/S data



Example of a user's timeline

New User Interface for Entering Episodes

http://l4all.dcs.bbk.ac.uk:8080 - L4ALL Episode Editor - Mozilla Firefox

← → ↻ × 🏠 🔍 Google 🔍 ABP

Edit Episode

1+2=3

— school —

Subject:

Qualification:

Nature:

Fact

Wish

Start:

End:

Title:

Description:

URL:

? **October, 1995** x

<< < Today > >>

| wk | Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|----|-----------|-----|-----|-----|-----|-----|-----------|
| 39 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 40 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 41 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 42 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 43 | 29 | 30 | 31 | | | | |

Select date

? **school**

Edit this episode

The fields in bold are compulsory.

Close

Done

Search for Timelines of “People like me”

- We encode timelines as ***token-based strings***, thereby allowing ***string similarity metrics*** to be applied to compare the user’s own timeline with other users’ timelines. Here are two examples:

| | | | | | | | | | |
|-----------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| S1 | Cl-6.4- 4.1 | Wk- J.0-2.1 | Un-9.1- 6.3 | Wk- R.0-2.4 | Wk- C.0-3.5 | Wk- P.0-2.3 | Un-6.4- 6.1 | Wk- G.0-4.2 | Wk- K.0-3.1 |
| S2 | Cl-0.0- 4.1 | Wk- R.0-2.4 | Un-6.4- 6.3 | Wk- N.0-9.2 | Un-6.4- 6.1 | Wk- S.0-3.1 | Wk- J.0-3.1 | Wk- J.0-2.1 | |

- Each token appearing in the above strings includes:
 - the category of the episode e.g. Cl for a College educational episode, Wk for a Work occupational episode, Un for a university educational episode
 - the primary and secondary classifications, to the depth of classification selected by the user for the matching process (up to a maximum of 4 – see later); in the above examples, a depth of 2 has been selected

Search for Timelines of “People like me”

- For the version of the system that we evaluated, four different similarity metrics were deployed:
 - Jaccard Similarity
 - Dice Similarity
 - Euclidean Distance
 - NeedlemanWunsch Distance
- These are all part of the SimMetrics Java package – see www.dcs.shef.ac.uk/~sam/stringmetrics.html and the Appendix for details (optional)
- A dedicated interface for the new search for “people like me” facility was designed, providing users with a three-step process for specifying their own definition of “people like me”:

Search for Timelines of “People like me”

1. The user specifies which attributes of their profile should be matched with other users' profiles, which acts as a filter of possible candidate timelines
2. The user specifies which parts of their own timeline should be matched with other users' timelines, by selecting the required categories of episode
3. The user selects the “depth” of episode classification that should be taken into account when matching their own timeline data with that of others (recall that each classification hierarchy may have up to four levels); the user also selects which of the four similarity metrics should be applied:

Search for people ...

... who share

- my age (within years)
- my gender
- my qualification
- my occupation
- my location

User Profile

Select the criteria that you want to find in the other people's profile.

... and who have a similar pattern with

- All Episodes
 - Learning Episodes
 - Attended school
 - Attended college
 - Attended university
 - Attended a particular course
 - Obtained a degree or diploma
 - Occupational Episodes
 - Personal Episodes
 - Other Episodes

Timeline

Check/uncheck the episodes you want to be considered in the pattern.

... using

Classification at level:

- 0 1 2 3 4

Search Method

Search Options

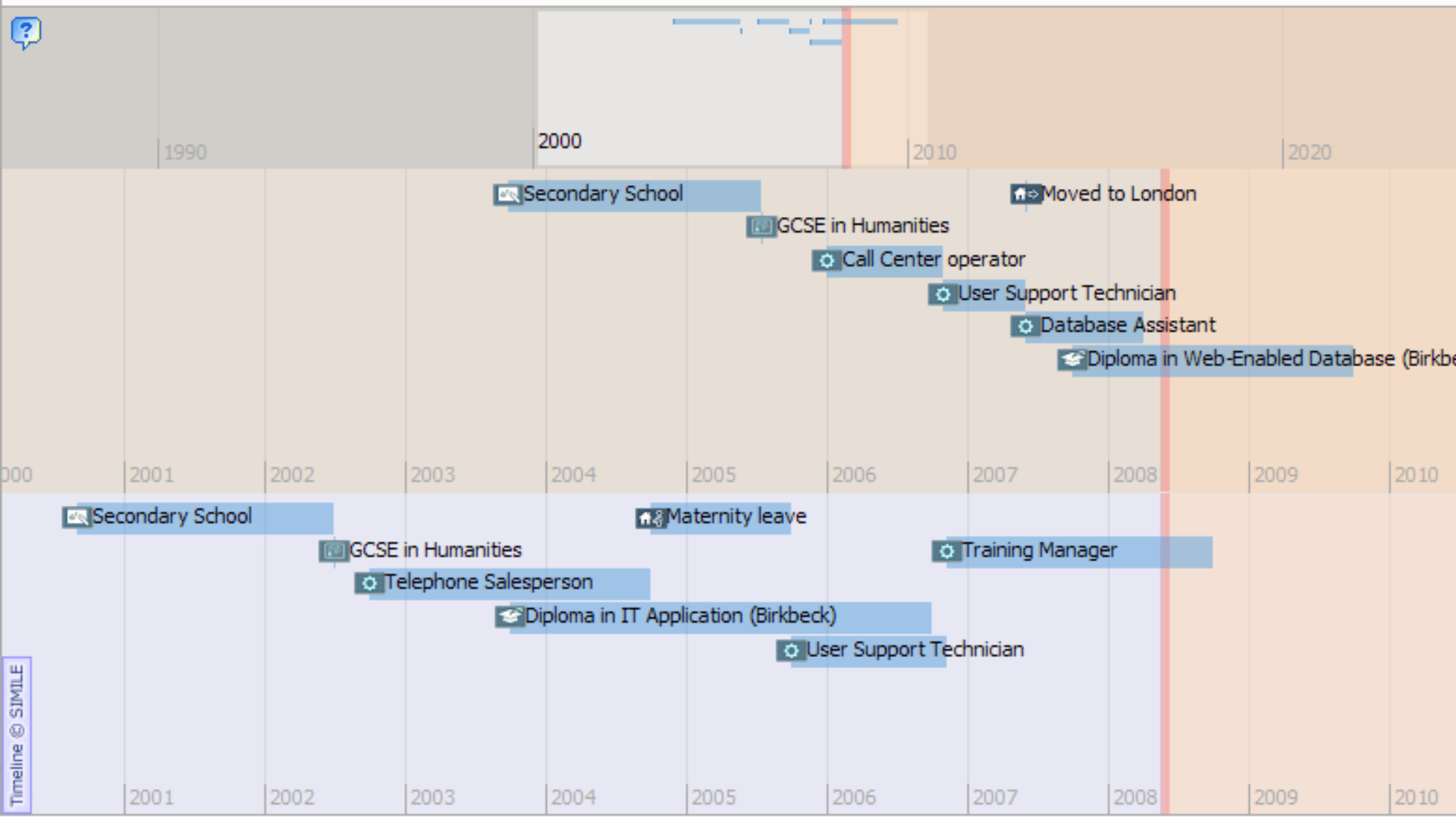
Select the criteria that you want to use to search for people like you.

Search for Timelines of “People like me”

- Once a definition of “people like me” has been specified, the system returns a list of all the candidate timelines, ranked by their normalised similarity
- The user can then select any of these timelines to visualise and explore
- The selected timeline is shown in the main page as an extra strip below the user’s own timeline
- The user’s and the selected timeline can be synchronised by date, at the user’s choice
- Episodes within the selected timeline that have been designated as being public by its owner are visible, and the user can select and explore these:

Timeline Filters

Filter: Highlight by keyword: Highlight by category:



Evaluation II

- The aim of this first design of a personalised search for “people like me” was to gather information about usage and expectation from users of such functionality
- An evaluation session was undertaken with a group of mature HE learners at Birkbeck, organised around three activities:
 - Activity 1: A usability study of the new system
 - Activity 2: An evaluation of the new searching for “people like me” functionality, focusing on participants exploring different combinations of search parameters and reporting on the usefulness of the results returned by the system
 - Activity 3: A post-evaluation questionnaire and discussion session

Evaluation II

- Activity 2 in particular required a significant amount of preparatory work, due to the need for an appropriate database of timelines to search over
- We opted for an artificial solution: providing participants with an *avatar*, i.e. a ready-to-use artificial identity, complete with its profile and timeline, and generating beforehand a database of other timelines based on various degrees of similarity with these avatars
- Further details of the evaluation session and the participants are given in our ECTEL'09 paper (optional)

Evaluation Outcomes

- Activity 1 indicated overall satisfaction with the main functionalities of the system; it also identified a number of usability issues most of which were subsequently addressed
- Activity 2 did not fulfil our expectations of identifying user-centred definitions of “people like me”:
 - Most participants took this activity at face value, selecting some parameters, exploring one or two of the timelines returned, and starting again
 - They could see no reason to try different combinations of search parameters, as their first try was returning relevant results
 - Participants could not easily see the benefit of this functionality i.e. why is it useful to find someone like me?

Evaluation Outcomes

- Two factors seem to have had a negative impact on this activity:
 - artificialness of the database used for the search (not enough variability in the timelines)
 - difficulties in grasping the meaning of some of the search parameters, notably the classification level and the search method
- However, during the discussion in Activity 3, it became apparent that participants could appreciate what this functionality could deliver if it were applied in a real context:
 - Namely, allowing the user to explore what other people with a similar background have gone on to do in their education and work episodes, thus identifying possible future choices for the user's own learning and professional development
- Also, every episode in timelines the participants produced was in fact correctly classified

Evaluation Outcomes

- In follow-on work, we have adopted just one similarity metric (NeedlemanWunsch) and we have explored a more contextualised usage of timeline similarity matching
- The information from the alignment between the two timelines is used to indicate, using different colours, the status of each episode in the target timeline:
 - blue for episodes that have a match in the user's own timeline;
 - grey for episodes judged to be irrelevant;
 - orange for episodes with no match in the user's own timeline: such episodes potentially represent ones that the user may be inspired to explore or even consider for their own future personal development

NeedlemanWunsch similarity metric

- This defines the distance between two strings of tokens X and Y to be the minimum cost of transforming X to Y by means a series of edit operations. The algorithm builds a cost matrix incrementally by considering a pair of tokens from each string (X_i and Y_j below) and the cost so far. The final cost at (X_n, Y_m) is the edit distance, where n is the number of tokens in X and m the number of tokens in Y:

$$Nw(X_i, Y_j) = \min \begin{cases} Nw(X_{i-1}, Y_{j-1}) + d(X_i, Y_j) & \text{substitution or copy} \\ Nw(X_{i-1}, Y_j) + G(X_i, Y_j) & \text{insert} \\ Nw(X_i, Y_{j-1}) + G(X_i, Y_j) & \text{delete} \end{cases}$$

- Here, $d(X_i, Y_j)$ is an arbitrary distance function between two tokens (typically 0 if $X_i = Y_j$ and 1 otherwise), and G is a function indicating the cost of a "gap" in one of the strings (typically 1).

Computing timeline alignments

- The system generates a similarity matrix between the two timelines (see below)
- Once the matrix has been completed, the potential alignments between episodes can be computed by tracking the Needleman-Wunsch computation and determining the "path" that led to the final (unnormalised) distance value between the two timelines.
- Moving along the path indicates how the episode tokens from both strings are matched:
 - A right move indicates no matching for the token in S2 and hence its alignment with a gap in S1;
 - A down move indicates no matching for the token in S1 and hence its alignment with a gap in S2;
 - A diagonal move indicates a match between the two tokens

| | | Target's Timeline (S2) | | | | | | | | |
|----------------------|------------|------------------------|------------|------------|------------|------------|------------|------------|------------|------------|
| | | | CI-0.0-4.1 | Wk-R.0-2.4 | Un-6.4-6.3 | Wk-N.0-9.2 | Un-6.4-6.1 | Wk-S.0-3.1 | Wk-J.0-3.1 | Wk-J.0-2.1 |
| User's Timeline (S1) | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | CI-6.4-4.1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | Wk-J.0-2.1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 8 |
| | Un-9.1-6.3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 9 |
| | Wk-R.0-2.4 | 4 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| | Wk-C.0-3.5 | 5 | 6 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| | Wk-P.0-2.3 | 6 | 7 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| | Un-6.4-6.1 | 7 | 8 | 7 | 8 | 9 | 8 | 9 | 10 | 11 |
| | Wk-G.0-4.2 | 8 | 9 | 8 | 9 | 10 | 9 | 10 | 11 | 12 |
| | Wk-K.0-3.1 | 9 | 10 | 9 | 10 | 11 | 10 | 11 | 12 | 13 |

NeedlemanWunsch similarity metric

- For example, an the alignment of these two timelines:

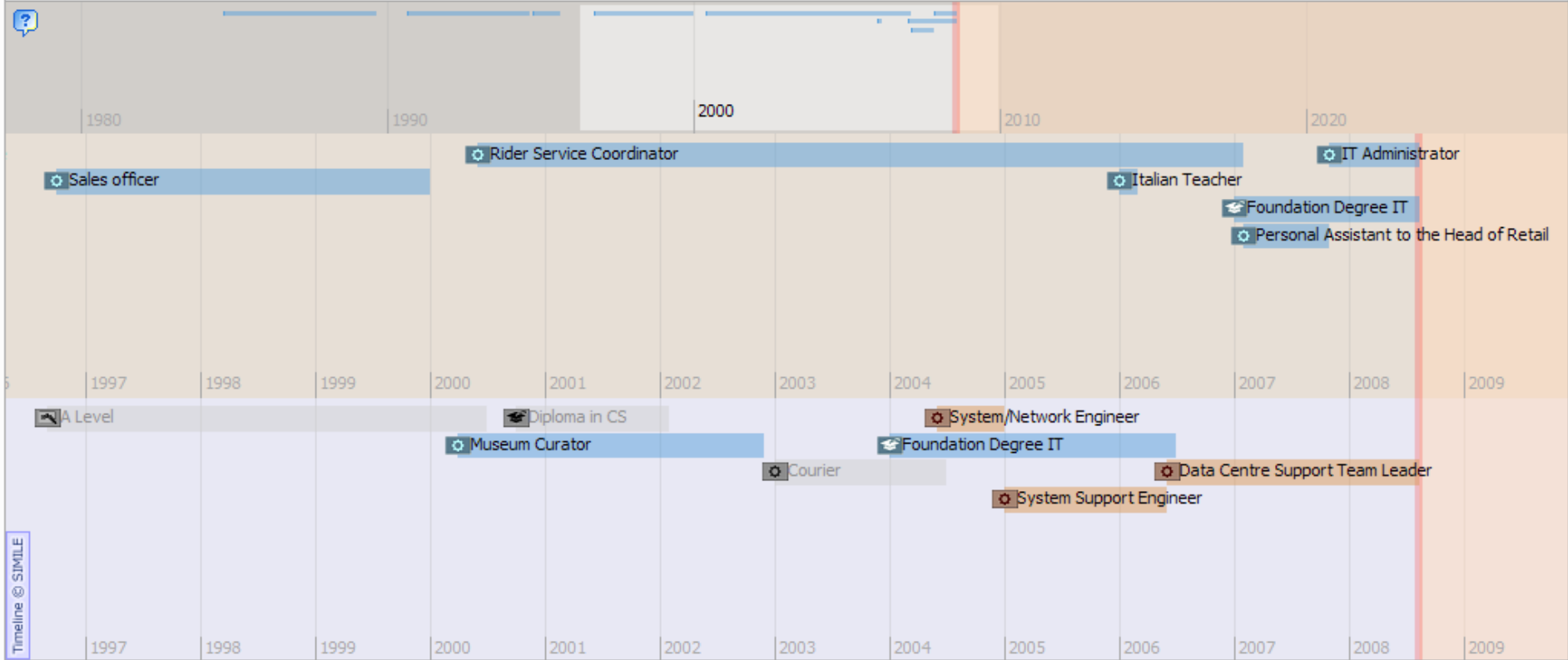
| | | | | | | | | | |
|-----------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| S1 | Cl-6.4-4.1 | Wk-J.0-2.1 | Un-9.1-6.3 | Wk-R.0-2.4 | Wk-C.0-3.5 | Wk-P.0-2.3 | Un-6.4-6.1 | Wk-G.0-4.2 | Wk-K.0-3.1 |
| S2 | Cl-0.0-4.1 | Wk-R.0-2.4 | Un-6.4-6.3 | Wk-N.0-9.2 | Un-6.4-6.1 | Wk-S.0-3.1 | Wk-J.0-3.1 | Wk-J.0-2.1 | |

- is derived from the path traced in the matrix above, and is as follows:

| | | | | | | | | | | | | | | | |
|-----------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|--|------------|------------|------------|
| S1 | Cl-6.4-4.1 | Wk-J.0-2.1 | Un-9.1-6.3 | | Wk-R.0-2.4 | Wk-C.0-3.5 | Wk-P.0-2.3 | | Un-6.4-6.1 | Wk-G.0-4.2 | Wk-K.0-3.1 | | | | |
| S2 | | | | Cl-0.0-4.1 | Wk-R.0-2.4 | | | Un-6.4-6.3 | Wk-N.0-9.2 | Un-6.4-6.1 | | | Wk-S.0-3.1 | Wk-J.0-3.1 | Wk-J.0-2.1 |

Timeline Filters

Filter: Highlight by keyword: Highlight by category:



Additional explanation of the timelines' alignment

About this user ...



alumni1

A Level (CI-0.0-4.1)

Museum Curator (Wk-R.0-2.4)

Diploma in CS (Un-6.4-6.3)

Courier (Wk-N.0-9.2)

Foundation Degree IT (Un-6.4-6.1)

System/Network Engineer (Wk-S.0-3.1)

System Support Engineer (Wk-J.0-3.1)

Data Centre Support Team Leader (Wk-J.0-2.1)

Match

You



Diploma in Commerce/Informatics (CI-6.4-4.1)



Software programmer (Wk-J.0-2.1)



Diploma in Architecture (Un-9.1-6.3)



Sales officer (Wk-R.0-2.4)



Rider Service Coordinator (Wk-C.0-3.5)



Italian Teacher (Wk-P.0-2.3)



Foundation Degree IT (Un-6.4-6.1)



Personal Assistant to the Head of Retail (Wk-G.0-4.2)



IT Administrator (Wk-K.0-3.1)



Lecture Summary

In this lecture we have discussed:

- Ontologies
- How ontologies are developed, including looking at two Case Studies
- Reasoning over ontologies
- Usage Scenarios for Ontologies in Information Systems
- Using Ontologies to support Personalisation, looking in particular at two Case studies – the SeLeNe and the L4A// systems

Question for you to consider: What are some of the similarities and some of the dissimilarities between these two systems?

Reading for this week

- Use cases for Ontologies in Information Fusion. M.M. Kokar et al. International Conference on Information Fusion, 2004, pp 415-421. At <http://www.fusion2004.foi.se/papers/IF04-0415.pdf> [read up to the end of Section 4; Section 5 is optional reading; read Section 6]
- Personalisation services for Self e-Learning Networks, K.Keenoy et al. International Conference on Web Engineering (ICWE'2004), 2004. At <http://www.dcs.bbk.ac.uk/selene/reports/SeLeNe-Personalisation.pdf> [you can skip sections 2 and 3]
- Searching for "People like me" in a Lifelong Learning System. N. Van Labeke et al. 4th European Conference on Technology Enhanced Learning (ECTEL'2009). At <http://www.dcs.bbk.ac.uk/~ap/pubs/ectel09VLMPFull.pdf> [you can skip section 5]

Other Background Reading (for interest only)

- “Ontology Development 101: A Guide to Creating Your First Ontology”. Natalya F. Noy and Deborah L. McGuinness. At <http://www-ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html>
- “Ontology”. Tom Gruber, preprint of an article appearing in the *Encyclopedia of Database Systems*, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag, 2009. At <http://tomgruber.org/writing/ontology-definition-2007.htm>
- SeLeNe project Deliverable 2.2: “Self e-Learning Networks - Functionality, User Requirements and Exploitation Scenarios”, 2003. At <http://www.dcs.bbk.ac.uk/selene/reports/Del22.pdf>
- SeLeNe project Deliverable 4.2: “Personalisation and Trails in Self e-Learning Networks”, 2004. At <http://www.dcs.bbk.ac.uk/selene/reports/Del4.2-2.1.pdf>
- MyPlan project Deliverable 4.1: “Report on the development of version 1 of the Personalisation Engine”, 2008. Available from <http://www.lkl.ac.uk/research/myplan/>
- MyPlan project Deliverable 4.3: “Report on the development of version 2 of the Personalisation Engine”, 2008. Available from <http://www.lkl.ac.uk/research/myplan/>

Appendix : The four similarity metrics used in L4All (for interest only)

All the similarity metrics assume two strings of tokens X and Y.

Needleman – Wunsch Distance. As defined earlier in the notes.

Euclidean Distance. The Euclidean distance creates two n-dimensional vectors from the strings X and Y. The vector space is the combined set of tokens of X and Y, and the distance is computed from the term counts X_n and Y_n (i.e. the number of occurrences of each token of the combined space in X and in Y, respectively):

$$Euclidean(X, Y) = \sqrt{\sum_n (X_n - Y_n)^2}$$

Jaccard Similarity. This considers the strings X and Y as sets of tokens and is defined to be the size of the intersection of the two sets divided by the size of the union of the two sets.

Dice Similarity. This also considers the strings X and Y as sets of tokens and is defined as twice the number of tokens in common between X and Y divided by the total number of tokens in X and Y.

Unlike the other three metrics, Needleman-Wunsch is able to detect non-similarity between a string and a permutation of it. Needleman-Wunsch is therefore useful in situations where the ordering of the tokens in the string is significant for the user. Euclidean distance is not able to discriminate between reordered strings, but does take into account the number of occurrences of a token in a string. Jaccard and Dice similarity are not able to discriminate between reordered strings and also do not take into account the number of occurrences of a token in a string. The Dice similarity gives higher similarity scores than Jaccard in cases where there is a small proportion of tokens in common between X and Y.