D4.1 Report on the development of version 1 of the Personalisation Engine

Project Information			
Project Acronym	MyPlan		
Project Title	MyPlan - Personal Planning for Learning throughout Life		
Start Date	1st September 2006	**End Date**	30th November 2008
Lead Institution	Birkbeck College, University of London		
Project Director	Prof. Alex Poulovassilis & Dr George Magoulas School of Computer Science and Information Systems, Birkbeck College, University of London, Malet Street, London WC1E 7HX, UK {ap,gmagoulas}@dcs.bbk.ac.uk		
Project Manager & contact details	Dr Nicolas Van Labeke Birkbeck College, University of London London Knowledge Lab 23-29 Emerald St London WC1N 3QS, UK nicolas@dcs.bbk.ac.uk		
Partner Institutions	Institute of Education; Community College Hackney; UCAS; Linking London Lifelong Learning Network		
Project Web URL	www.lkl.ac.uk/research/myplan/		
Programme Name (and number)	e-Learning Capital Programme		
Programme Manager	David Kernohan		

Document Name			
Document Title	D4.1 - Report on the development of version 1 of the Personalisation Engine		
Reporting Period	April 2007 – December 2007		
Author(s) & project role	Nicolas Van Labeke (project manager – WP4 leader)		
Date	01/10/2007	**Filename**	MyPlan D4.1.doc
URL			
Access	☒ Project and JISC internal	☐ General dissemination	

Document History		
Version	**Date**	**Comments**
0.1	11/10/2007	First draft ready for revision
1.0	20/11/2007	First draft ready for revision
2.0	21/04/2008	Final document circulated

Executive Summary

This deliverable covers workpackage WP4 (Development and deployment of personalised functionalities for planning of lifelong learning) and reports on the current state of the (re)design of the L4*All* system. The developed personalisation engine will provide: (i) personalised search of timelines from "people like me"; (ii) personalised recommendation of which course(s) to study next; (iii) customisation of the delivery and presentation of contents.

The design of the system is now reaching completion and is under internal testing before being transferred to the server that will be used for the online evaluation and user testing. This report was initially planned at month 12 but has been postponed to month 14. That was necessary in order to cover the extra work (2 months) required for the redesign of the user interface, which was not included in our original project plan. The main activities performed to achieve our targets in this workpackage included:

- Redesign of the GUI of the L4*All* system, using DHTML/javascript for the front-end and JSP/servlet for the back-end.
- Redesign of several aspects of the ontology underlying the L4*All* system to accommodate for the new functionalities: different categories of user (learner, expert, institution), a two-axis taxonomy of events, etc.
- Design and implementation of a similarity measure engine for the comparison of learner's timelines. The mechanism is based on converting timelines into string of comparable tokens and on using string metrics for ranking them.
- Design and implementation of a recommendation engine, using the timeline formalism to represents requirements/recommendations and the similarity engine for proposing matches.
- Design and deployment of several customisation procedures (colour/shapes used in the timeline visualisation, bookmarks for interesting timelines, etc.)

Table of Content

Figures and Tables

1 Introduction

As outlined in deliverable D3.1 (MyPlan Personalisation Specification), the personalisation functionalities intended for L4*All* and prioritised for the current development of the system as are follows:

1. Personalised search of timelines from "people like me"
2. Personalised recommendation of which course(s) to study next
3. Customisation of the system
4. Automatic update of users' profiles
5. Ability to record and display ratings of search results

The current version of the L4*All* system, as described in this document, is tackling the first three issues, and the automatic update and user-defined rating will be addressed in the future. The reasons for the focus on the first three issues is that they required a significant amount of development, as well as the need for a redesign of the Graphical User Interface.

This document is organised as follows. First, the new GUI will be presented, highlighting the significant changes both at presentation and functionality levels. Second, we will address the changes in the back-end engine that were required in order to support the new functionalities. Third, the similarity metrics used for comparing timelines and the essential elements for performing personalisation in L4*All* will be discussed. Fourth, the personalisation engine, encompassing the search for similar timelines, the personalised recommendation and the customisation, will be presented.

2 Redesign of the Graphical User Interface

The deployment of the new personalisation functionalities in the L4*All* system suffered some delays from the work plan initially devised. The main reason was that the original Graphical User Interface used for the creation and manipulation of a learner's timeline (initially developed as a Flash module, embedded in the JSP-powered web pages) didn't offer the necessary flexibility to incorporate additional functionalities. A decision was therefore made to redesign the interface of the system, using DHTML and javascript-powered widgets to deploy the timelines. This decision allowed us to improve the possibilities of learners' interaction with their timeline (maintenance of their own and, more important, visual comparison between their timeline and someone else's) and therefore to better support personalisation of timeline-based functionalities.

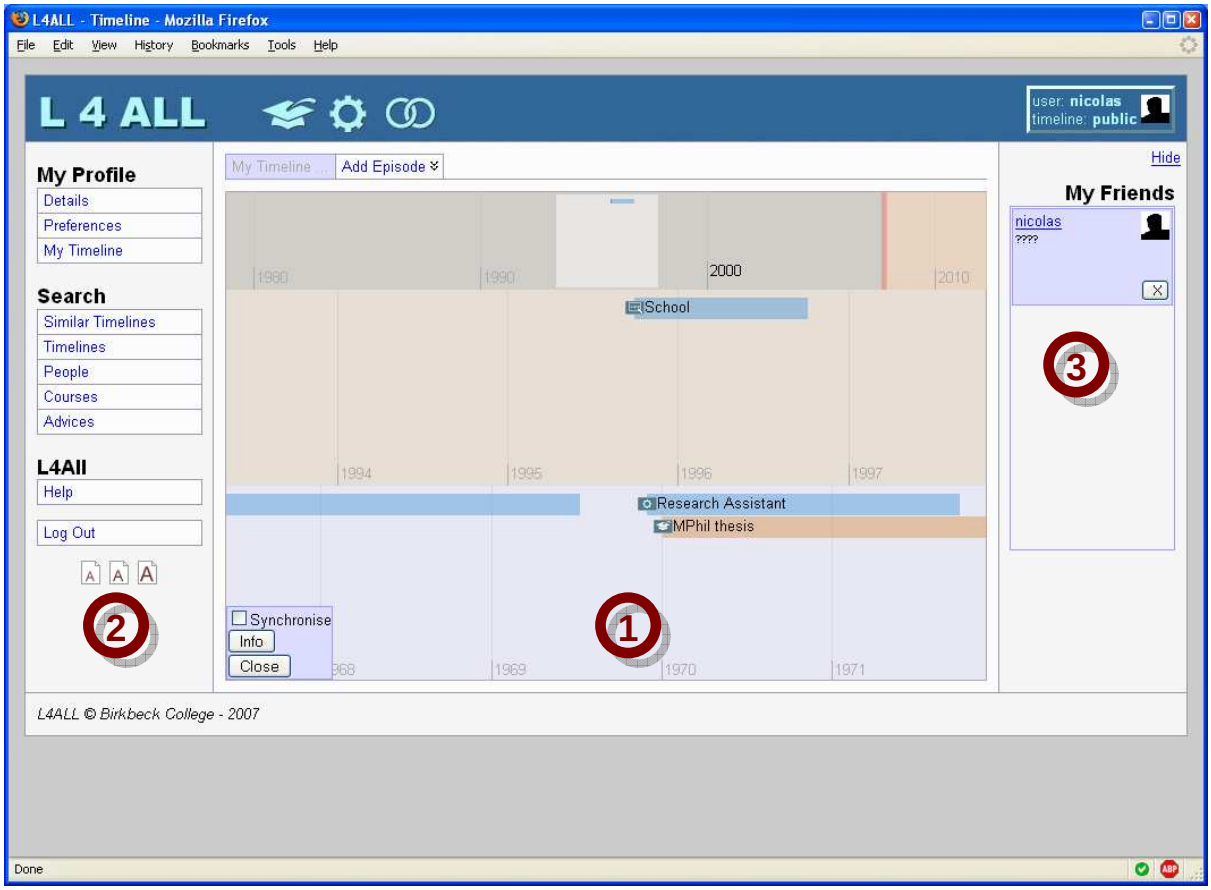


Figure 1. An overview of the L4*All* main interface.

Figure 1 shows a snapshot of the main page of the current system. At the core of the system is the representation of the user's timeline (labelled ① in the figure), around which all functionalities of L4*All* are now organised: user profile and timeline management, keyword search for courses in the LearnDirect database, keyword search for people and timelines registered in the L4*All* system, search for similar timelines, search for pathways recommendations, etc. Access to these functionalities is provided by specific items in the left-hand side menu (labelled ② in the snapshot). On the right-hand side of the timeline lies a bookmarks space (labelled ③ in the figure) used by the learner to store shortcuts to interesting timelines (gathered by searching for people or "similar" timelines, see below).

2.1 Visualisation of Timelines

The visualisation of the timeline is supported by a dedicated DHTML widget[1]. It represents every episode in a user's timeline, in a chronological fashion (see Figure 2). The timeline space is divided in two zones[2]:

- The large strip at the bottom contains the whole timeline in "real-size", with a time scale (typically in years) that allows a clear visualisation of every event.
- The small strip at the top presents a summary of the timeline with a smaller time scale (usually in decades), which provides a quick overview of the whole timeline (or at least a

[1] The SIMILE timeline is a DHTML-based AJAX widget for visualizing time-based events, see http://simile.mit.edu/timeline/

[2] A third strip can also be present, for displaying another timeline in parallel to the user's one, see below, section 2.3.2.

significant chunk).

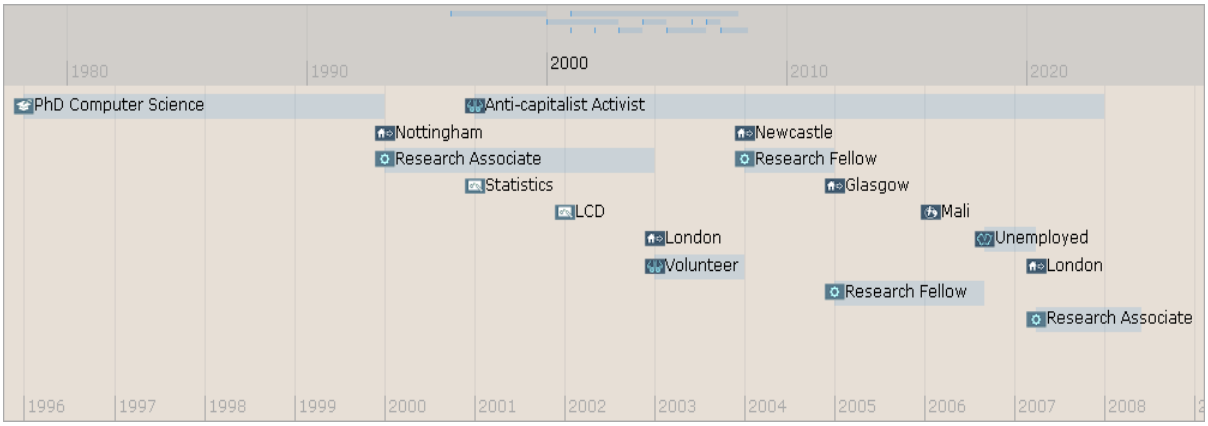


Figure 2. An example of a lifelong learner's timeline, as visualised in L4*All*.

Both strips can be dragged backward and forward, showing past and future episodes. Both strips are also synchronised, meaning that dragging one will re-centre the second accordingly. This mechanism allows an easy and intuitive manipulation of the timeline by its user. Episodes are accessible to the user: by simply clicking on one of the block in a strip, a "balloon" help pops up (see Figure 3), containing more detailed information about the selected episode (dates, description), as well as access to edit/deletion functions (see below).

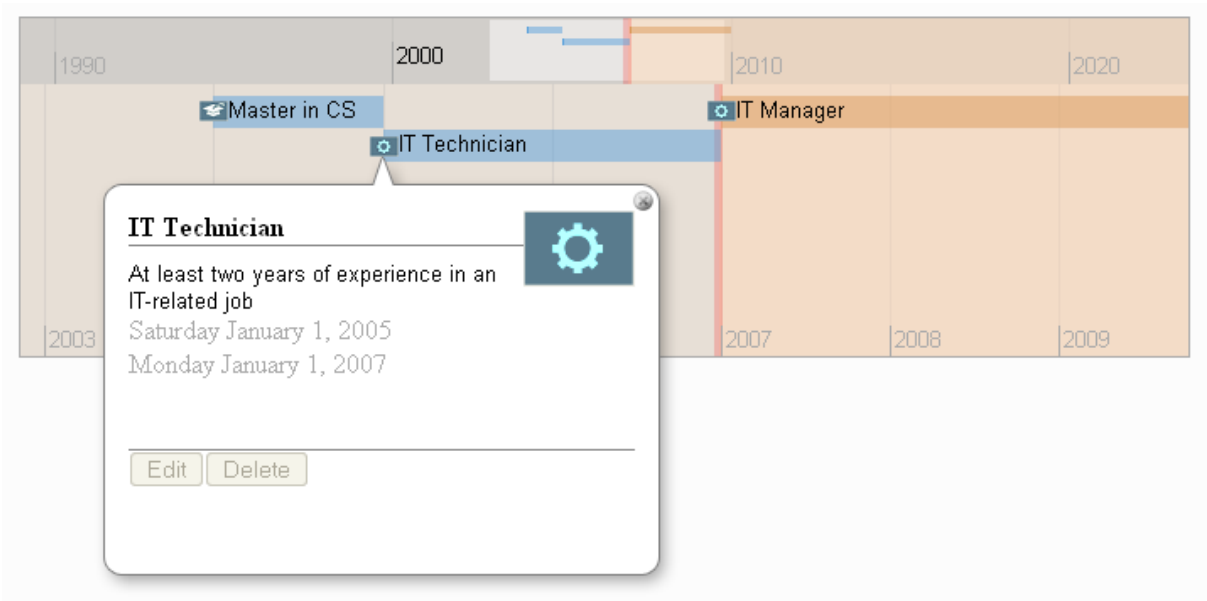


Figure 3. Visualising the properties of an episode.

Each episode is represented in the timeline by an icon specific to its type (work, school, travel, etc.) that locates its start date, by its title and by a block representing its duration (if applicable). A colour scheme is also used to visualise the difference between factual episodes (in blue) and desired ones (in orange)[3]. The timeline itself uses different

[3] One of the first user-testings of the new interface however indicates that a colour scheme would be more useful if used to differentiate the type of an episode (e.g. blue for occupational episodes, orange for professional, green for personal); a different graphical mechanism would be used for the fact/desire distinction.

background colours to represent the past, the future and the current date (as a light red bar crossing the strips). Most of these visualisation parameters can be modified by the user, using the customisation facility offered by L4*All* (see Figure 4). The scale of the main strip can be modified (for example by switching to monthly or yearly increment of the timeline), as well as the colours used to emphasise different aspect of a timeline (background colour for the past and future, filling colours for factual and desired episode)[4].

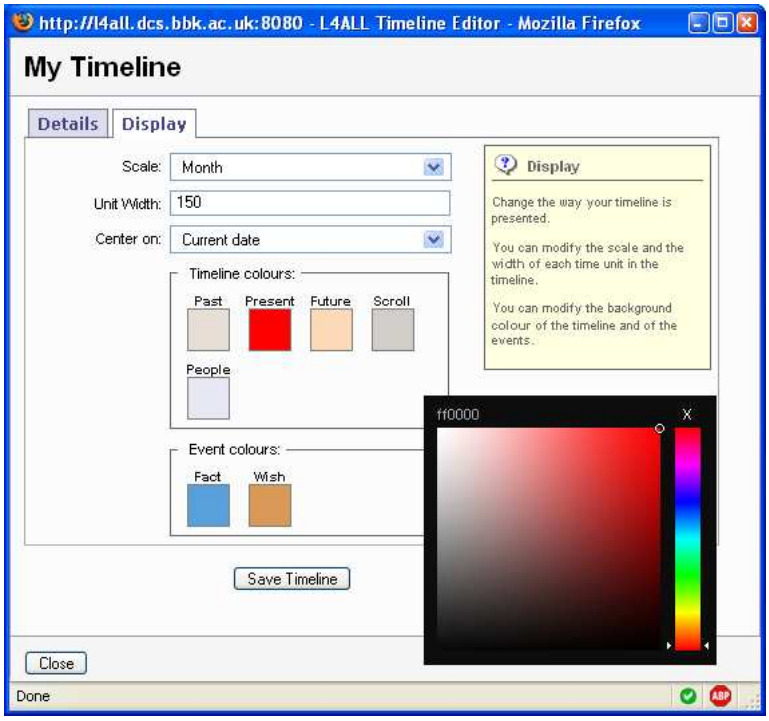


Figure 4. Customisation of the L4*All* user interface, where the visualisation of the timeline can be personalised.

*2.2 Accessing L4*All *Functionalities*

As mentioned above, the redesign of the interface was not only to address some serious technical limitations but also to present a different usage paradigm. In the previous version of L4*All* (see deliverable DC1 of the L4*All* project), a Flash-based interface presented the users with some alternatives: visualising their timeline, searching for people, and searching for other timelines. What we are trying to achieve with the current version is to put an emphasis on the key element of the project, the timeline manipulation. In essence, every functionality offered to the users beside the timeline manipulation is a satellite of it. Practice reinforces that feeling, for example by allowing the result of a course search to be added to the user's timeline.

The current version of the system maintains all the previous functionalities (search for people, timelines and courses by keyword) and provides access to the new ones (search for similar timelines, enquire for recommendation). All functionalities are available in the left-hand side menu in Figure 1 and are deployed in a popup window that maintains a connection with the main page of the web site. By doing so, the central role of the timeline is maintained, while still providing sufficient support to users for performing other tasks.

All the popup windows used to support secondary tasks have the same layout (see Figure 5 for a dialog box that allows users to modify their preferences). Most windows in the system

[4] It has to be noted that several other aspects of the timeline could be customised. The current selection is a trade-off between time and proof-of-concept.

are form-based, using basic HTML widgets like edit box, drop-down box, radio and check boxes, etc. to support editing of the required information, including buttons for validating the form or closing the window[5].

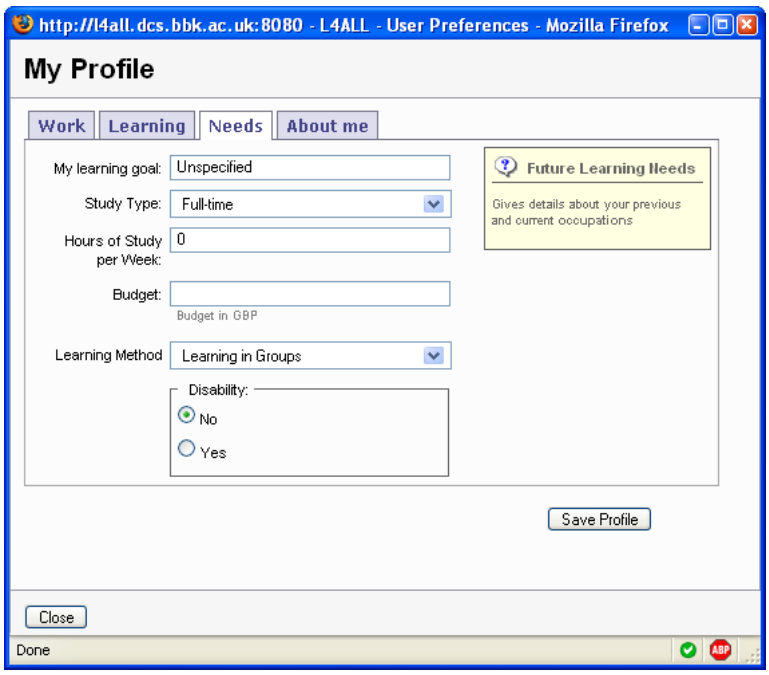


Figure 5. Modifying the user profile and preferences.

All the forms in L4*All* have a dual validation process. On the client-side, an AJAX script is used to check, prior to validation, the syntactic validity[6] of the most important fields in the form: well-formed dates, well-structured email, existence of required information, etc. Then, upon validation on the server-side, the appropriate servlet verifies the semantic validity of the submitted information: password is correct for the registered user, episode existence before modification or suppression, etc.

In order to limit the confusion arising from too many popup windows opening on the user's screen, we deliberately limited their proliferation by enforcing one single external window. If the user activates another L4*All* functionality from the main menu, the URL of the popup window – if open – is modified to point to the new document. Some of the functionalities are supporting one-off actions (such as profile modification) but others involve a more or less complex sequence of actions (such as searching for timelines, which results in visualising the result and handling individual timelines, see Figure 6). In such a case, all the steps to accomplish the functionality take place sequentially in the popup window, maintaining the possibility of navigating back one step using the browser's history.

[5] The first user-testing indicated a serious issue with validating or closing the form. The validation button, as can be seen in the figure, usually lies below the form, whereas the close button, used to shut down the window, was initially put in the footer of the form. This layout caused some confusion, leading users to close the window by mistake. Corrections will be made accordingly.

[6] Using Zapatec AJAX Form scripts (http://www.zapatec.com/), which are a fast and easy way to perform validation, provide feedback, and display error messages that enrich the user's experience while reducing the communications with the server behind the scenes.

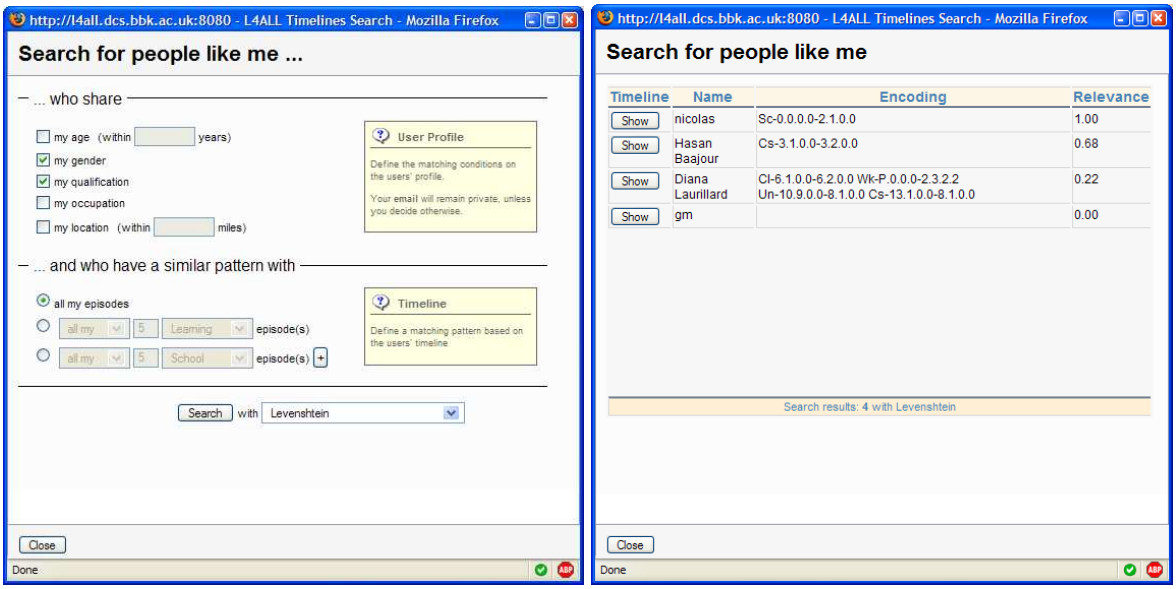


Figure 6. The query interface for searching for "people like me" (left) and the result of the search (right).

2.3 Feeding Back to the Timeline

The most important feature of the new design of the L4*All* interface is clearly the deliberate attempt to *integrate* the various functionalities with the core of the system, i.e. the timeline. Two aspects are particularly considered here: the exploitation of the search for particular courses and the exploration of timelines of other users.

2.3.1 Exploiting Courses

As in the previous version of L4*All*, the user has the possibility to query databases for appropriate courses, both internally (within the dedicated L4*All* course repository, used mainly for evaluation purposes) and externally (using a remote connection to the LearnDirect service). The result of a search is presented to the user, who has the possibility to explore an individual course (see Figure 7, right). Upon finding a course they consider interesting, users can include it in their timeline.

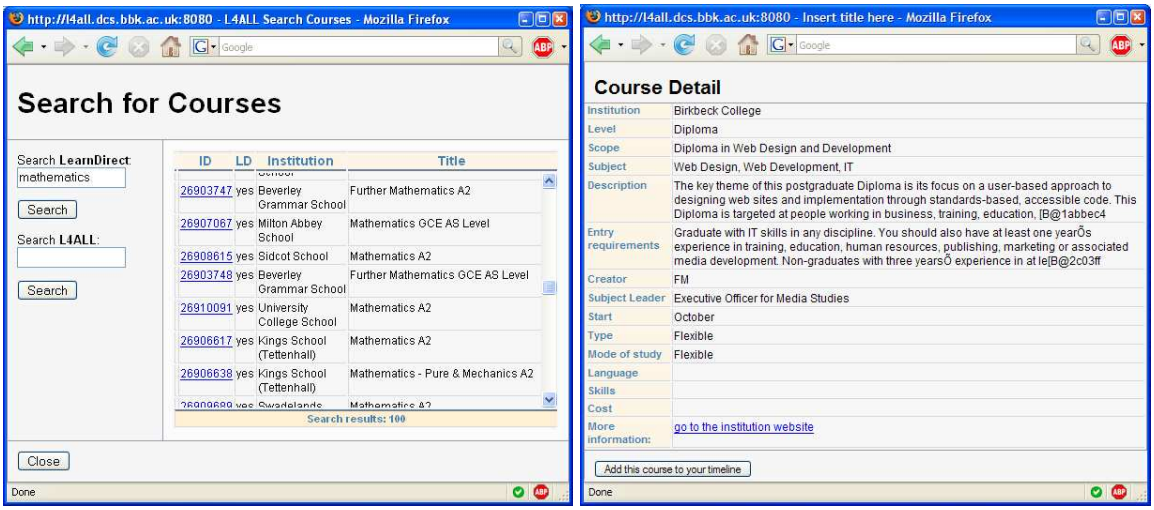


Figure 7. Searching for courses and exploring their details.

This action opens the "Create a New Episode" dialog box, where the necessary information (dates, description, URL) can be added and/or modified (see Figure 8). Note that this inclusion does not maintain any formal link with the courses database, as it consists of

creating a whole new episode whose content is *extracted* from the course description. If the course is later on removed from the database, the episode remains in the timeline.

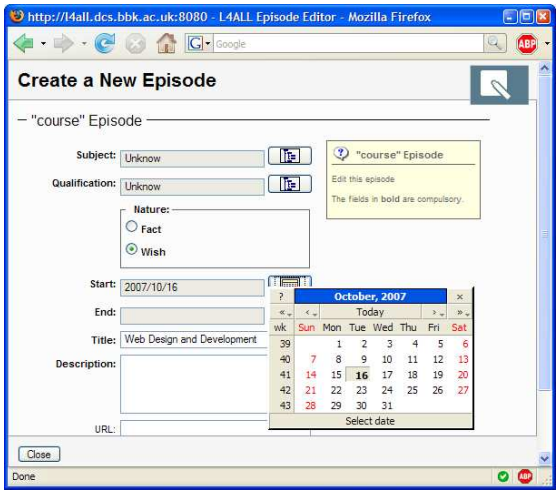


Figure 8. Adding the selected course in the user's timeline.

2.3.2 Exploiting Timelines

When users are searching for other people or timelines (either performing a keyword-based search or using the similarity-based “people like me” feature, see Figure 6), the outcome is a list of other timelines that they can explore for their own purpose (e.g. inspiration for future pathways, identification of a role model, "expert" background, etc.). In the previous version of the system, the visualisation of another user’s timeline was separated from the visualisation of the user's own timeline, i.e. it was taking place through a separate interface. As a result, it didn’t allow users to *simultaneously* view two timelines and compare them. However, one of the reasons for accessing someone else's timeline would be to find interesting or motivating episodes to incorporate into your own timeline. There is, therefore, a strong incentive for allowing users to visualise several timelines at the same time, an activity that the new timeline widget clearly supports.

When prompted to show someone else's timeline (for example by clicking on an individual items after having searched for “people like me”, see Figure 6), the corresponding timeline is incorporated in the timeline visualisation widget as an extra strip (see Figure 9). Both the user's and extra timelines can be synchronised by date or remain in their original form, at the user's will, depending on the situation. Public episodes in the extra timeline are visible, meaning that users can explore them and, ultimately, integrate them in their timeline[7].

[7] An intuitive way for doing this integration would be by dragging-and-dropping episodes, a feature that the current timeline widget does not support. Expanding it would be too demanding resource-wise, so an indirect method has been implemented: a new episode is created, as when adding a course in the user's timeline, which is initially filled with information from the original episode.

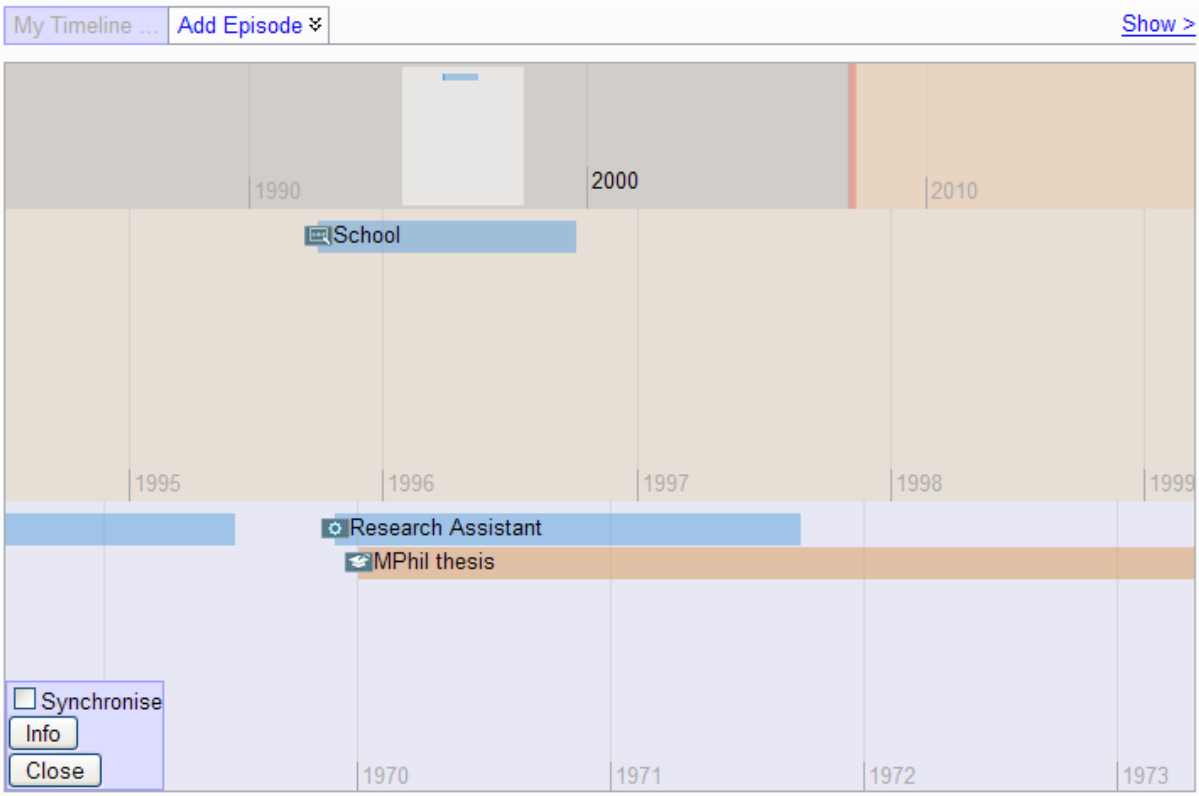


Figure 9. Displaying the user's timeline (top) in parallel to another user's timeline (bottom).

2.4 The Administrative Desktop

The last addition to the new interface is an administrative desktop that gives access to restricted, password-protected parts of the system. The main restricted function of the system is the ability to create special types of users. The default access to the system is for "standard" users, lifelong learners who are given support in managing their educational and professional pathways. MyPlan is introducing two other types of user:

- "Expert users" have exactly the same access to the system functionalities as "standard" users but their timelines reflect essential pathways and are important for other users to explore.
- "Recommendation users" are artificial users whose timelines act as templates for the recommendation engine (see below, section 5.2).

Because of their sensitive nature, the creation of these two types of user is not available from the main interface but is accessible through the administrative desktop (see Figure 10).

Other functions available in the administrate desktop include user management (listing, editing and deletion), access to the L4*All* course system (adding new courses to the database, deletion and edit) and the dedicated editor for the recommendations (see below).

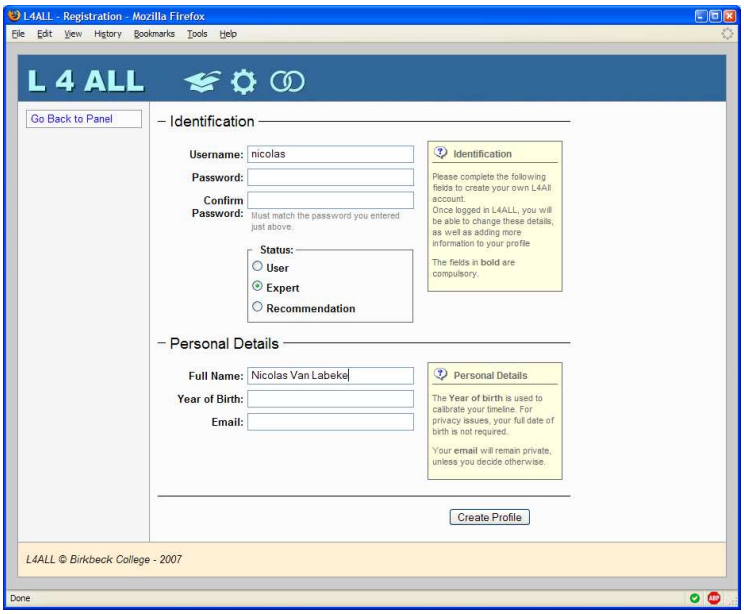


Figure 10. Registering an expert user in the administrative desktop.

3 Modifications of the Back-End Engine

The changes at the back-end of the system fall in four categories: the overall architecture, the user model and the episode and timeline models.

3.1 System Architecture

The overall system architecture remains the same, as documented in previous deliverables of the L4*All* project (see for example L4*All* Deliverable D6.2 and Figure 11).

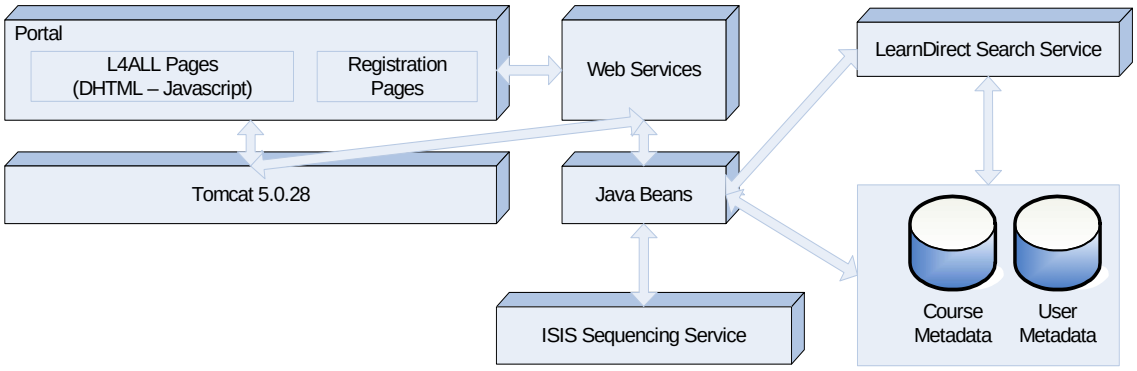


Figure 11. L4*All* System Architecture.

Changes were introduced at the level of servlets in order to operate a clearer distinction between the client-side and the server-side of the L4*All* system. All servlets have been separated into two sets (see Figure 12):

- The *Service Servlets* are an implementation of the different web-based services provided by L4*All* (e.g. GetTimeLineDetails, GetCourseDetails, etc.). Their inputs have been normalised and output certified as proper XML documents.
- The *Interface Servlets* are an implementation of the server-side response to the different parts of the L4*All* interface and are basically called from the relevant JSP documents.

Part of this clearer distinction means that some of the functionalities of L4*All* are associated with two distinct servlets, one for the server-side service, outputting an XML document

containing the result of the request (for example *SearchSimilarServlet* is handling queries for searching for “people like me”) and one for the client-side interface generation (for example, *ProcessSearchSimilar* is gets the search parameters typed by the user in the query form, calls the *SearchSimilarServlet* to retrieve the XML document, parses this document and generates an appropriate web-page). A partial flowchart, representing the most important functionalities of the L4*All* system, can be seen in Figure 13.

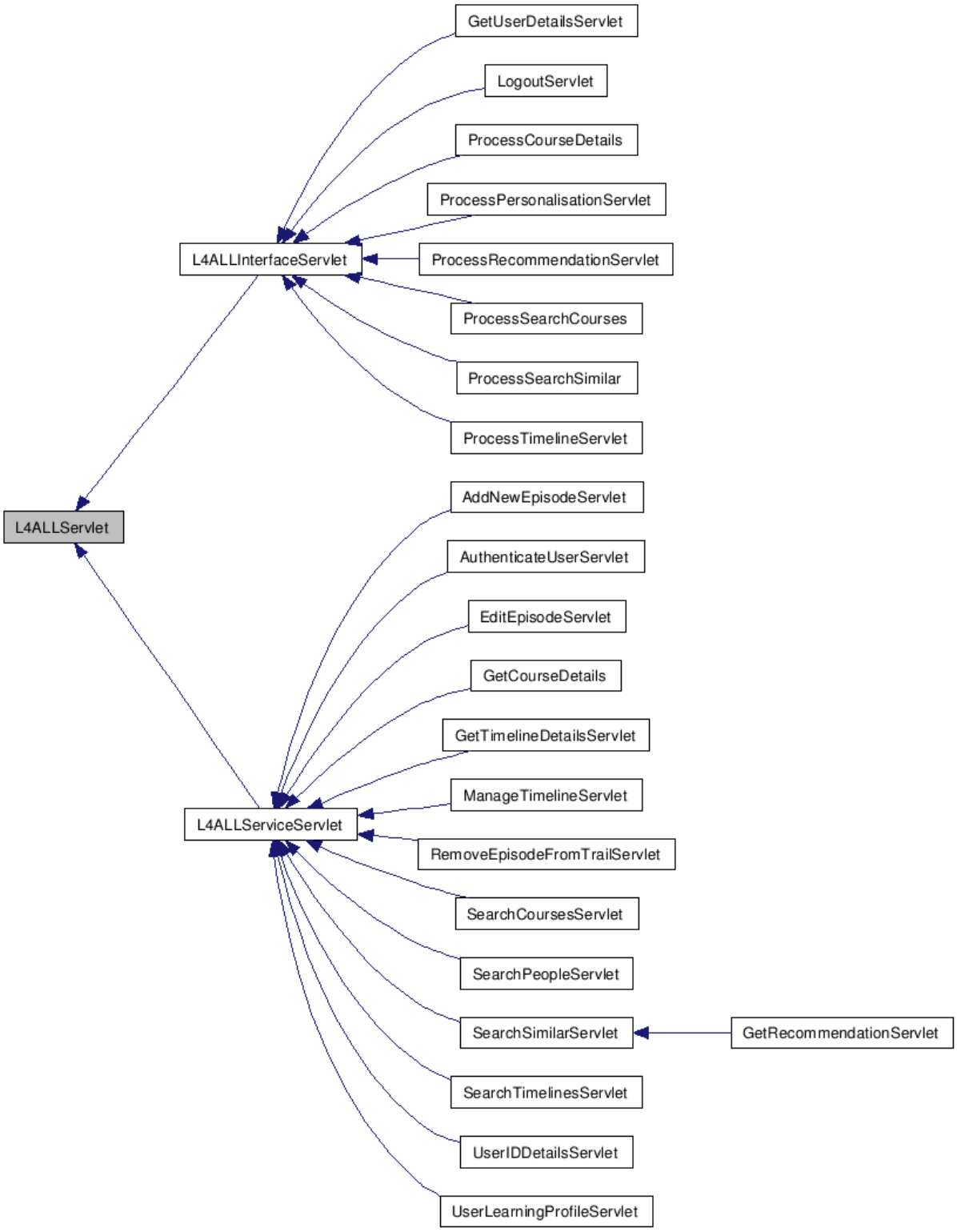


Figure 12. The Servlets (interface and service) of the L4*All* system.

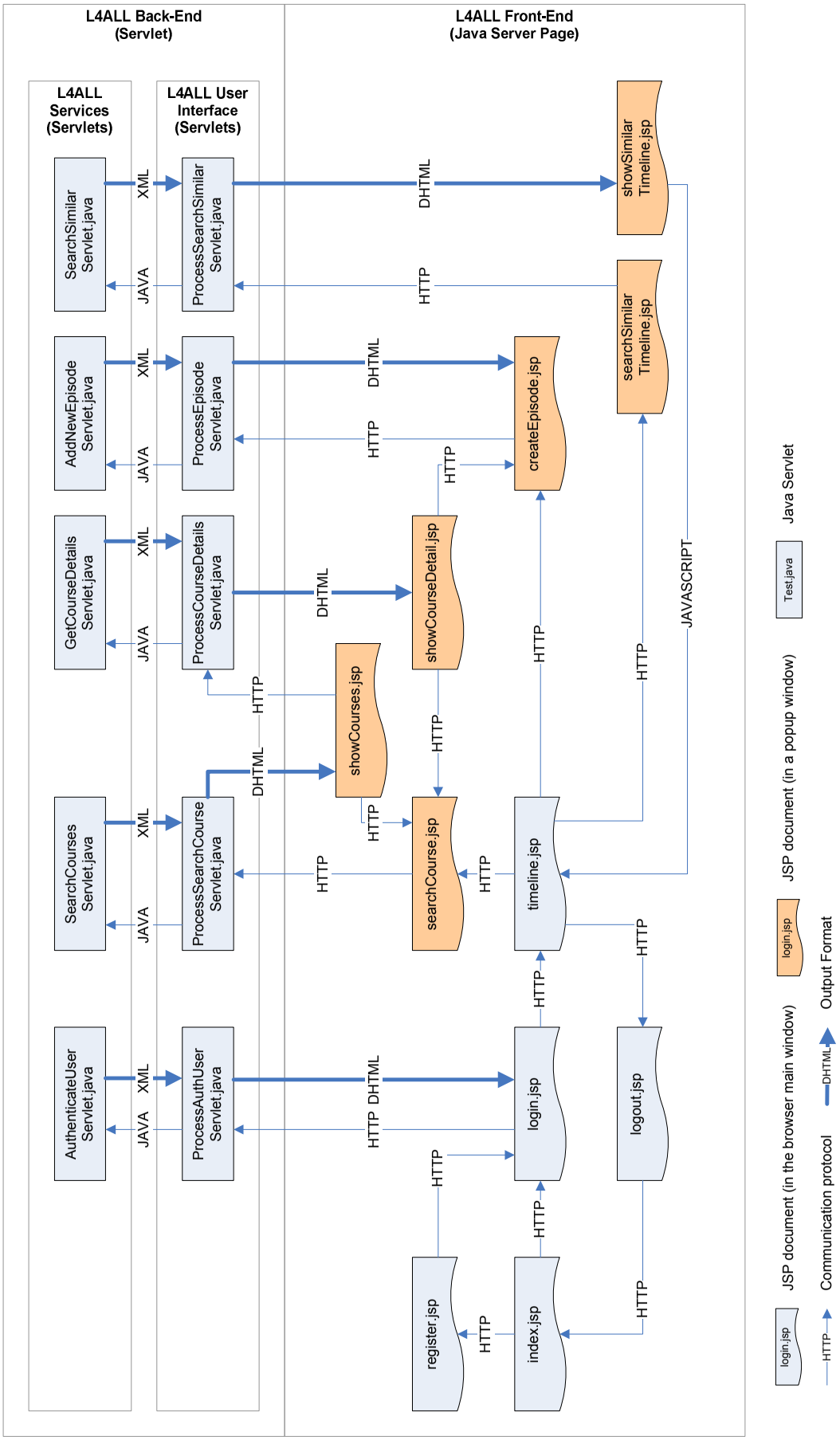


Figure 13. An extract of the flowchart of the L4*All* system.

3.2 User Model

The following properties have been added to the RDF definition of a user model:

- **Status:** indicates if the current model represents a "standard" user (value: "user"), an expert (value: "expert") or a recommendation provider (value: "template").
- **Year of Birth:** the age of the user is now deprecated, both for ethical and practical reasons. L4*All* is now using the year of birth of a user to inform lower and upper limits of a timeline (value: four-digit integer).

3.3 Episode Model

The following properties have been added to the RDF definition of an episode:

- **Status:** indicates whether an episode in a timeline is considered by the user as a fact (value: "fact") or a desire (value: "wish"). Note that in the case of a recommendation, this property takes the meaning of prerequisite and aspiration (see below).
- **Primary Classification:** contains the identifier of a single element, locating this episode within the first classification (value: a four-part string "#.#.#.#", see below).
- **Secondary Classification:** contains the identifier of a single element, locating this episode within the second classification (value: a four-part string "#.#.#.#", see below).

The definition of episodes in L4*All* has also been refined (see Table 1):

- Introducing an explicit separation between personal, professional and educational episodes
- Normalising the (unique) identifier of each type of episode
- Specifying whether episodes have a duration or are considered instantaneous.

Four distinct classifications[8] have been used in L4*All* to further specify the exact nature of some of the episodes:

- Education episodes are specified primarily by a subject (SBJ) and secondarily by a qualification level (NQF).
- Work/Voluntary episodes are specified by an industry sector (SIC) and an occupation/position (SOC)
- Business episodes are specified by an industry sector (SIC).

At the current stage, it is assumed that episodes of other types do not require any further classification as their nature does not justify any further discrimination[9]. As much as possible we have maintained the structure and identifiers of each of the above taxonomies but, for deployment purposes, we have limited their depth to four levels. Consequently, each element in the taxonomies can be uniquely represented by four-digit identifier (see Figure

---

[8] As much as possible, we intended to use standard classifications, whether governmental or institutional. The (non definitive) choice for the current system is based on these four classifications: the *Standard Industrial Classification* (SIC), the *Standard Occupational Classification* (SOC), the *National Qualification Framework* (NQF) and the Labour Force Survey's *Subject of Degree* (SBJ). For details on the encoding and evolution of these standards, see the Labour Force Survey User Guide (http://www.statistics.gov.uk/downloads/theme_labour/Vol5.pdf).

[9] For example, we are assuming that disability episodes will be considered as a self-defined item, whatever the nature of the disability. The evaluation and a long-term usage should indicate whether such assumption can be maintained or finer distinctions are needed for particular types of episodes.

14), each digit uniquely identifying a precise sub-level in the classification “tree”.

Table 1. The different episodes available in the L4*All* system and their classification

ID	Type	Description	Duration	1st	2nd
Educational Episodes					
Sc	School	Attended school	✓	SBJ	NQF
Cl	College	Attended college	✓	SBJ	NQF
Un	University	Attended University	✓	SBJ	NQF
Dg	Degree	Obtained a degree		SBJ	NQF
Cs	Course	Attended a particular course	✓	SBJ	NQF
Occupational Episodes					
Wk	Work	Employed	✓	SIC	SOC
Vl	Voluntary	Voluntary work in charity/voluntary organisation	✓	SIC	SOC
Bs	Business	Setup a business	✓	SIC	
Ml	Military	Attended military service	✓		
Re	Retired	Retired			
Ue	Unemployed	Unemployed	✓		
Cr	Carer	Homecarer	✓		
Personal Episodes					
Mv	Moved	Moved to a different location			
Tv	Travel	Spent some time abroad	✓		
Ch	Child	Birth in the family			
Ad	Adoption	Adopted a child			
De	Death	Death in the family			
Ma	Married	Got married			
Se	Separated	Divorced			
Ds	Disability	Developed a (permanent) disability			
Il	Illness	Developed a (temporary) illness	✓		
Other Episodes					
Ot	Other	Any user-defined episode not covered previously	✓		

```
<xml version="1.0" encoding="UTF-8"?>
<list>
   <item>
      <label>Unknown</label>
      <attribute name="id">0.0.0.0</attribute>
   </item>
   <item>
      <label>NQF Level 6</label>
      <attribute name="id">7.0.0.0</attribute>
      <list>
         <item>
            <label>Bachelor Degree (with Honours)</label>
            <attribute name="id">7.1.0.0</attribute>
         </item>
         <item>
            <label>Graduate Certificate</label>
            <attribute name="id">7.2.0.0</attribute>
         </item>
         <item>
            <label>Graduate Diploma</label>
            <attribute name="id">7.3.0.0</attribute>
         </item>
      </list>
   </item>
</list>
```

Figure 14. Extract of the XML encoding of the qualifications classification.

3.4 Timeline Model

The following property has been added to the RDF definition of a timeline:

- **Status:** as with the user's status above, indicates if this timeline belongs to a "standard" user (value: "user"), an expert (value: "expert") or a recommendation provider (value: "template").

4 Similarity Metrics

The initial prototype of the L4*All* system supported several search functionalities over users and their timelines. Two limitations of this approach were identified during the first piloting phase. First, all the search functionalities were keyword-based, targeting the various fields of the User Profile, Learning Profile and Timelines, and therefore limited in their scope. In particular, searching on timelines returns matches based solely on the occurrence of the keywords present in one or several episodes but cannot exploit the overall structure of the timeline. Second, the results of any search were not personalised according to the particular user performing the search. An alternative approach was needed, that could take into account both these issues: in other words, some form of comparison or similarity measure between a user's timeline and the rest of the timelines in the L4*All* repository.

String metrics offer such a possibility. String metrics (also referred to as similarity metrics) have been widely used in information integration and in several fields of applied computer science, more rarely in the context of Intelligent Tutoring Systems, where they have been used in the REDEEM system [ref?] to compare alternative sequences of instructional activities as produced by authors.

4.1 Timeline Encoding

In the context of the L4*All* timelines, the main requirement for using similarity metrics is to encode a time-based sequence of records into a token-based string. For this purpose, we have made four simplifying assumptions at the outset (the implications of these assumptions for users will be explored in our forthcoming evaluation activities):

1. *The precise duration and dates of an episode have no particular significance.* This may seem strange for a time-dependent data structure but the relevance and usage of such information for searching for "people like me" is ambiguous. Should we consider two learners having done the same university degree but at different dates similar or not? Should we consider them more different if one of them has taken twice as long as the other (being part-time for example)? Or is it enough, at some level, to consider them similar since both of them have done this particular degree? In the absence of evidence supporting one point of view against the other, we decided, initially, to ignore this information. Only each episode's relative time-stamp (i.e. its position in time compared to the other episodes in the timeline) is used in an attempt to "linearise" the timeline by ordering the episodes in chronological order.
2. *Gaps between episodes have no particular significance unless explicitly expressed as an episode.* The problem posed by gaps in timelines is the lack of explicit explanation for their occurrence and therefore for their significance for the timeline. Again, in the absence of such information, they are ignored.
3. *Some categories of episode may have no role to play in defining "people like me"*. The purpose of a timeline is for learners to record every episode of their background that may have an impact on their learning pathways. For example, personal episodes such as marriage, illness, relocation, etc. are important as they may have a clear influence on the decisions made for personal development (e.g. a course at a particular learning institution may have been followed because of relocation to a particular city). However, this does not necessarily mean that such episodes are a prerequisite or a necessary condition for reaching a particular stage in someone else's development. Their

importance while searching for role models, inspiration, or “people like me” are therefore ambiguous and subjective. Therefore, whether to include or not particular categories of episode in the similarity matching should be left to the user to specify.

4. *The exact classification of an episode may not be significant in defining “people like me”.* As described earlier, some of the most important episodes in the timeline (educational and work-related episodes) use a specific attribute to precisely describe their instance, e.g. working as a researcher in computer science. However, taking such a fine-grained description of an episode might not be useful in searching for “people like me”, as it may make more sense to consider that a researcher (without a precise field) is someone that should be considered “like me”. Therefore the level of specialisation of episodes should also be left to the user to specify.

Using these assumptions, it is now relatively straightforward to generate a token-based string representing the timeline. Each episode of the timeline is encoded as a string token composed of a two-letter unique identifier of the category of the episode (e.g. **Cl** for a College episode, **Wk** for a Work episode, see Table 1) and two four-digit codes classifying the exact instance of this episode (as described in the previous section). Note that, in order to maintain a consistent pattern for the token's encoding, nonexistent or unspecified classifications are encoded as **0.0.0.0**.

Combining the two first assumptions mentioned above means that no time information is used to encode episodes, only their relative position matters[10]. Filters are then applied to the string of tokens to remove episodes that should not be considered in the current similarity search, as well as limit the depth of their classification. In the latter case, the use of the coding system for the classification facilitates that process: digits below the specified depth are replaced by 0, replacing the specific classification by a more general parent.

4.2 A Comparison of String Metrics

We have conducted a study over a set of string metrics that are part of the SimMetrics JAVA package; this is an open source extensible library of metrics that provides real number-based similarity measures between strings, allowing both normalised and un-normalised output[11]. The SimMetrics package contains about 20 different metrics, some of them customisable by using user-defined cost functions and tokenisers. Not all metrics are applicable in our context, since some are tailored for working on a particular application domain (linguistic for example) and require strings that are incompatible with our encoding of timelines.

Table 2 shows a set of synthetic timelines used in our comparison study. They are deliberately simplistic in their structure, as the purpose of this comparison was to identify general trends arising from the various similarity metrics, rather than evaluating their intrinsic power of discrimination. The Source timeline is a string of four episodes of different type: college (Cl00), university (Un00), move (Mv00) and work (Wk00). Each episode has been encoded as a token, using the scheme described in the previous section. For the sake of clarity, and since this comparison does not rely on the full power of discrimination of the scheme, the episode classifications have been reduced to a single digit each (i.e. representing 0.0.0.0 as 0).

The target timelines represent a variety of alterations of the Source timeline that could occur in real-life situations: a totally similar timeline (i.e. the same sequence of episodes), a

[10] With an arbitrary decision as to their ordering if multiple episodes coincide in time.

[11] The *SimMetrics* library (http://www.dcs.shef.ac.uk/~sam/simmetrics.html) is the Java package ultimately used in the project; tests have been also made on a similar project, *SecondString* (http://secondstring.sourceforge.net/)

reordered timeline (i.e. the same episodes but totally reordered), adding an extra episode, removing an existing episode, substituting an episode by another one. Note that the set of target timelines listed in the table only represent the most representative timeline of each group. In order to test the behaviour and consistency of the metrics, all possible combinations were generated for each group (e.g. timelines representing the addition of a new episode were generated considering every possible position in the Source timeline).

Table 2. List of encoded timelines used for the metrics comparison.

ID	Description	Encoding
Source	The original timeline used as the source for the similarity measure	Cl-00 Un-00 Mv-00 Wk-00
Id	A timeline similar to the source.	Cl-00 Un-00 Mv-00 Wk-00
Re	A timeline containing the same episodes as the source but in a totally different order (i.e. no episode is at the same position in the string).	Un-00 Wk-00 Cl-00 Mv-00
Ad_W	A new work episode (similar to an existing one) is added to the timeline.	Cl-00 Un-00 Mv-00 Wk-00 Wk-00
Ad_E	A new episode (different from all existing ones) is added to the timeline.	Cl-00 Un-00 Mv-00 Wk-00 Bs-00
RM_W	The last episode is removed from the source timeline.	Cl-00 Un-00 Mv-00
RM_U	One of the episodes of the source timeline is removed.	Cl-00 Mv-00 Wk-00
SB_E	One of the episodes of the source timeline is substituted by a new one (different from all existing ones).	Cl-00 Un-00 Mv-00 Bs-00
SB_U	One of the episodes of the source timeline is substituted by an existing episode.	Cl-00 Un-00 Mv-00 Un-00
SB_W	One of the episodes of the source timeline is substituted by a variant of an existing episode.	Cl-00 Un-00 Mv-00 Wk-10

4.3 Results and Interpretation

Table 3 summarises the results of the different similarity measures applied between the Source timeline and every target timeline. The values shown in the table do not represent the distance between the two strings, but rather their normalised similarity, i.e. the ratio between the calculated distance and the maximum distance. As mentioned earlier, the main aim of this comparison was not to focus on individual measures for assessing their accuracy but to extract general conclusions regarding their behaviour when confronted with particular configurations. From these results, several conclusions can be drawn. First, all the similarity measures are indeed able to recognise complete similarity between timelines (as indicated by all 1 in the ID column). More interestingly, three groups of metrics emerge, as listed in Table 3.

Table 3. The normalised similarity between the source and the test timelines.

	ID	RE	Ad_W	Ad_E	RM_W	RM_U	SB_E	SB_U	SB_W
Levenshtein	1	0	0.8	0.8	0.75	0.75	0.75	0.75	0.75
Needleman - Wunsch	1	0	0.8	0.8	0.75	0.75	0.75	0.75	0.88
Jaro	1	0.72	0.93	0.93	0.92	0.92	0.83	0.83	0.83
Matching Coefficient	1	1	0.8	0.8	0.75	0.75	0.75	0.75	0.75
Euclidean Distance	1	1	0.84	0.84	0.8	0.8	0.75	0.75	0.75
Block Distance	1	1	0.89	0.89	0.86	0.86	0.75	0.75	0.75
Jaccard Similarity	1	1	1	0.8	0.75	0.75	0.6	0.75	0.6
Cosine Similarity	1	1	1	0.89	0.87	0.87	0.75	0.87	0.75
Dice Similarity	1	1	1	0.89	0.86	0.86	0.75	0.86	0.75
Overlap Coefficient	1	1	1	1	1	1	0.75	1	0.75

The first group includes transformation-based metrics like Levenshtein, Jaro and Needleman-Wunsch that are able to discriminate between the basic operations of string manipulation (copy, substitution, addition, deletion). The non-zero result for the Jaro distance in the RE column can be explained by a threshold used for determining matching tokens (see the documentation of this metric in the Appendix); our test strings are not long enough (only four tokens) to allow proper discrimination. All these metrics do not take into consideration the position of the token involved in one of the string manipulations (whatever the location of the added or substituted episode, the scores are the same). The only exception is the Needleman - Wunsch distance, which gives a different score when a variant of the initial episode (i.e. same category but different classification) is substituted (score of 0.88 in SB_w, instead of 0.75 in SB_e and SB_u). This is due to the use of specific gap cost and distance functions that can be tailored to the particular nature of the data involved in the similarity measure and therefore could be adjusted for our particular use of the timelines.

The second group of metrics includes vector-based metrics such as Block Distance, Euclidean Distance and Matching Coefficient that are not able to discriminate between re-ordered strings, as indicated by 1 in the RE column. Whatever the order of the tokens in the string, both source and target are considered to be identical since they contain the same set of tokens. As with the metrics in the previous group, the results for addition, substitution and removal of tokens are position-independent.

The third group of metrics includes the rest of the vector-based metrics (Jaccard, Cosine, Dice Similarities and Overlap Coefficient) which, as with the previous group, do not discriminate between reordered tokens. Moreover, this group also fails to take into account the duplication of tokens in the string, as exemplified by the scores of 1 in the Ad_w column (i.e. adding an episode that is already existing in the timeline) or the different scores for the SB_u column (i.e. substituting an episode with one that is already existing, resulting in fact in the deletion of this episode). Once again, this is because of the set-based algorithms used for these metrics, in particular the use of intersection/union procedures rather than summation as in the previous group. This is also reflected by the fact that substitution also depends on the nature of the episode substituted (the SB_u column give scores different from the other substitutions). In this group, the Overlap Coefficient is an extreme case, as it basically measures whether the source string is a subset of the target one (or the converse).

What the comparison above shows is that different similarity metrics offer different degrees of support for the basic operations of string manipulation: copy, substitution, addition or deletion of a token. The important point here is that the comparison does not highlight one particular metric as being more useful or accurate for our purpose, precisely because our purpose (or, rather, the user's) is unknown. Our encoding assumptions encompass a wide range of users' behaviour regarding the way they perceive the “people like me” functionality.

5 The Personalisation Engine

The personalisation engine of the L4*All* systems is a set of functionalities that covers different features available to lifelong learners: searching for timelines of "people like me", searching for recommendations of what to do next and customisation of the system.

5.1 Searching for Timelines of “people like me”

In order to validate assumptions presented in section 4.1, a dedicated interface for the searches was therefore designed and implemented. It provides users with a three-step process for specifying their own definition of “people like me”. The first step of a user's query specifies those attributes of the user's profile that should be matched with other users' profiles (age, qualification, location, etc.) and acts as a filtering of the possible candidates before application of the similarity comparison on the timelines. The second step of the query specifies which part(s) of the timeline should be taken into account for the similarity comparison (currently by selecting the appropriate categories of episode). The final step

specifies the nature of the similarity measure to be used (i.e. depth of episode classification and metric). Once a definition of “people like me” has been specified by the user, the search returns a list of all candidate timelines, ranked by relevance (i.e. their normalised similarity measure). Users now have the possibility to access any returned timelines and explore them.

This first -- but mainly artificial -- approach for offering the “people like me” functionality gave us the possibility to accumulate information about usage and expectation from users. It has offered us some insights about the context and relevance of particular configurations and how specific aims -- like looking for inspirational timelines or learning recommendations -- should be supported.

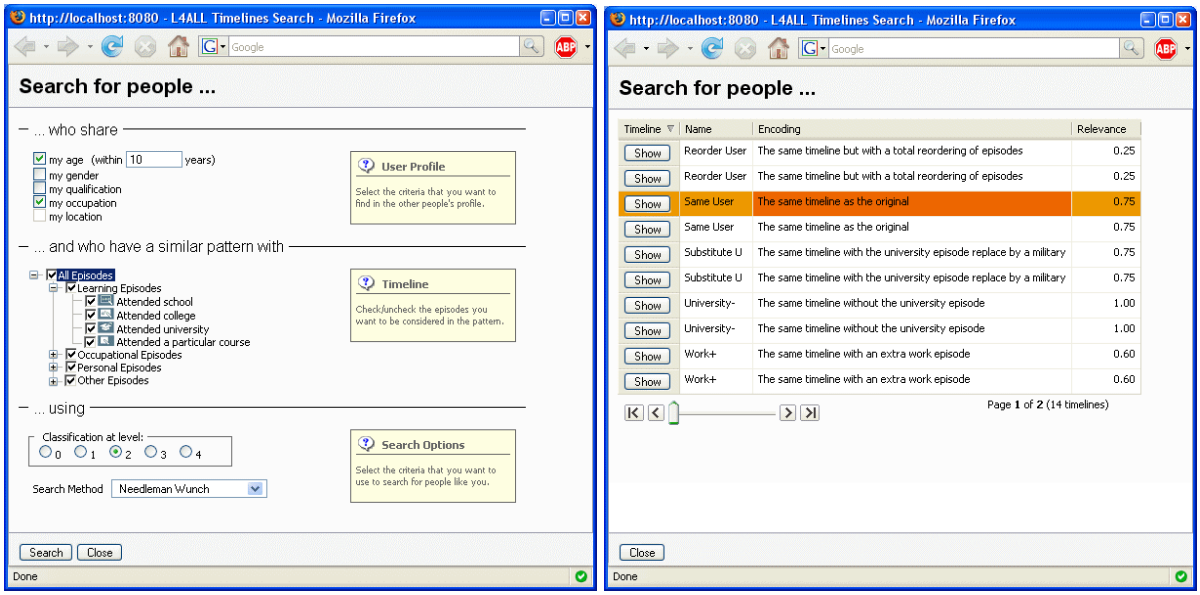


Figure 15. Searching for “people like me”.

A second – more rooted – approach was to redefine the existing search functionalities offered by the L4*All* system (i.e. keyword-based search with respect to users’ profiles and timelines). As mentioned before, these searches were not personalised: every search using the same keywords will return the same list of matches, whoever the user might be. The current redefinition of the search procedure removes the artificial distinction between searching with respect to users' profiles and users' timelines by providing a single interface. The results of the search, i.e. a set of matching timelines, are presented using the selected ranking criteria (i.e. one of the selected similarity metrics).

5.2 Recommendations

Some of the changes made in the L4*All* data structures (e.g. “template” timelines, fact vs. wish episodes, etc.) have been made in order to support a recommendation mechanism, allowing learners to explore and plan future (learning) activities.

The driving principle for designing a recommendation engine was to re-use the existing data structures, in particular the timeline. The hypothesis we made is to consider the timeline as suitable for representing both learners’ own pathways AND recommendations.

Recommendations are considered as a special case of timelines which, instead of representing the *concrete pathway* of a learner, represent an *abstract template of a possible pathway* that can be instantiated in a specific context.

Recommendations are built – like any other timeline – by adding episodes of a specific nature. But two aspects are fundamentally different:

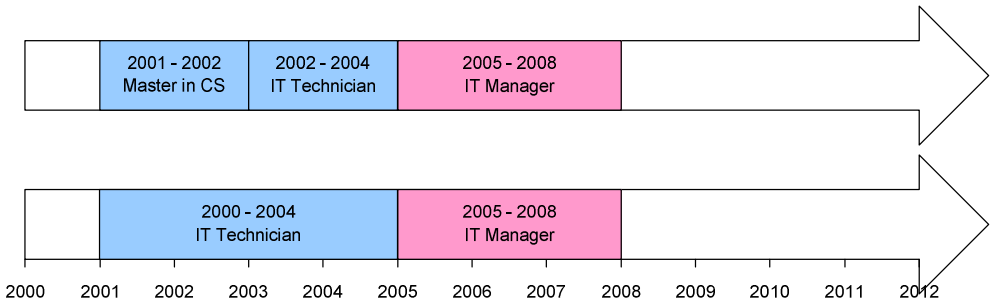
1- Episodes in a recommendation timeline are divided into two groups:

a. *Aims* which represent the goal or outcome of following that particular recommendation

b. *Prerequisites* which represent every mandatory step in achieving the aims.

2- Unlike a learner's timeline, time information is considered relative rather than absolute. Start dates are used to locate episodes in relation to each other (i.e. episode 1 HAS to take place before episode 2) while duration indicates the MINIMUM amount of time spent on a particular requirement.

Here are two examples of such recommendations, represented as timelines:



Both have for their *aim* getting a position as an **IT Manager** (represented by a **Work** episode, in pink). The first recommendation has two *prerequisites*: a **Master in Computer Science** (represented by a **University** episode, in blue) and two years of experience as an **IT Technician** (a **Work** episode, in blue). The second recommendation has only one prerequisite: 4 years of experience as an **IT Technician** (a **Work** episode, in blue).

As mentioned above, the dates used for each episodes, do not really matter in an absolute sense. The **Master in CS** episode, defined as starting in 2001, could have been defined at any other dates. What matters is that this episode is *two years* long and is taking place *before* the **IT Technician** one.

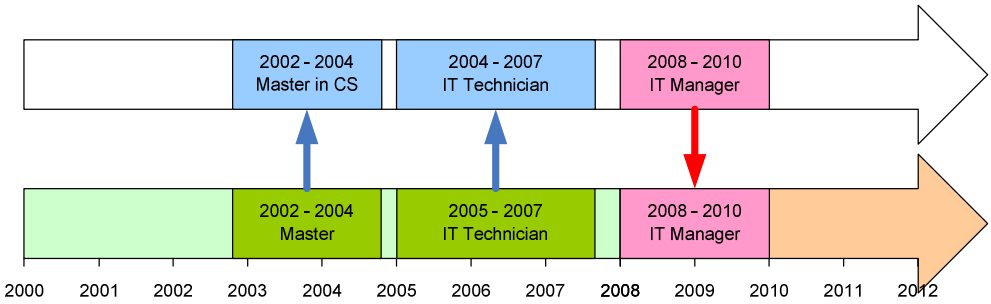
*Note that the aim of both recommendations (**IT Manager**) is likely to be open-ended, as there is no significance attached to how long such a position should last, but it is displayed with a duration for the sake of uniformity.*

Put together, the idea of these recommendations is to express the fact that, in order to become an IT manager, one has to have either 4 years of experience or a Master in Computer Science and two years of experience.

The engine starts computing the match between a learner's timeline and each recommendation. This match is based on the similarity distance – as used in the search for "people like me" functionality – between the timeline and the *prerequisites* of the recommendation. *Only the prerequisites of the recommendation timeline are used in the matching process, as we are looking for relevant experience or requirement to propose a relevant pathway.*

The idea is for the learner to inspect one recommendation at a time, by displaying it *simultaneously with his own timeline*. By doing so, the *abstract template* that the recommendation represents is *instantiated* in the context of his timeline, by matching every relevant episode.

In the case of the first recommendation, both requirements are met, i.e. matched with similar episodes in the learner's timeline. The aim of the recommendation, once inspected by the learner can be extracted and proposed as a *wish episode* to be added in the *future* part of his timeline.



Such matching and aligning of timelines is made possible by the use of one particular set of distance metrics, the Needleman – Wunsch metrics. This metric is measuring similarity between two token-based strings by calculating the minimal number of changes (i.e. adding a token, removing a token, and modifying a token) necessary to transform one string into the other:

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

where *d* is an arbitrary distance function between two tokens (typically returns 1 if the tokens are identical, 0 if not) and *G* the gap cost function.

More important, during this matching process, it also builds a transformation matrix that can be used to backtrack the computation and find the *longest common subsequence* between the two strings.

The Needleman-Wunsch algorithm finds the optimal alignment of a pair of sequences by optimising a score function, that is, each possible alignment is scored according to a score function, and the alignment that yields the highest score is the optimal alignment of pair of sequence. If the score of more than one of the possible alignments equals the highest score, there is more than one optimal alignment of the pair of sequences.

The algorithm builds a substitution matrix that represents the cost of editing one string into the other, using the three basic operations:

- Copy character from string1 over to string2 (cost d)
- Substitute one character for another (cost d)
- Delete a character in string1 (cost G)
- Insert a character in string2 (cost G)

Let's assume we have a learner's timeline encoded as **AJC** (i.e. containing episode **A**, **J** and **C** in this order) and a recommendation encoded as **ABCN**. The result of running the similarity metric on both strings can be seen in Figure 16. The final score of 2 (in the bottom-right corner of the table) indicates that a minimum of 2 edit operations are needed to transform one string into the other.

By backtracking the edit distance computation (materialised by the lines in Figure 16), two possible alignments are found (the **-** character indicates a gap in the string):

AB-CN **A-BCN**

A-JC- **AJ-C-**

The two alignments came from the fact that, after aligning the tokens A, the next edit operation can be inserting B then J (in blue) or inserting J then B (in red).

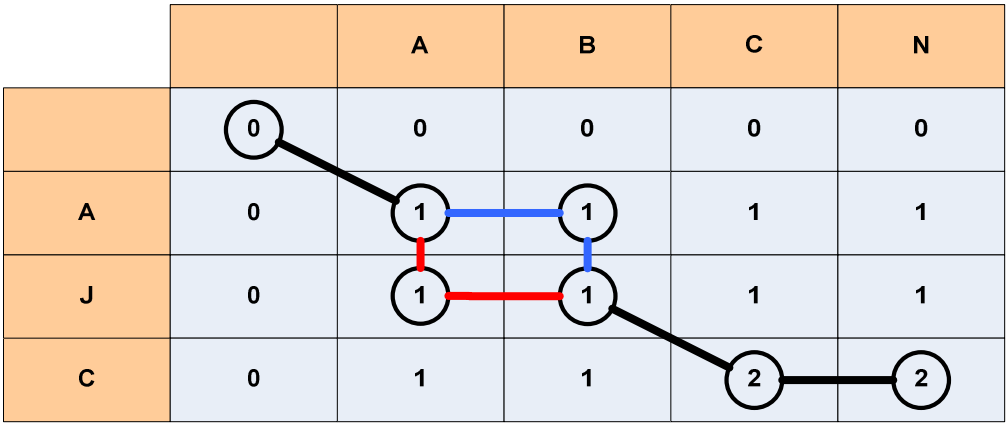


Figure 16. The Needleman – Wunsch distance matrix.

These alignments tell us four things:

1. The *longest common subsequence* (i.e. the sequence of tokens common to both strings) is **AC .**
2. How the recommendation's abstract *timeline* can be instantiated and mapped to the learner's timeline (i.e. by matching the elements of the longest subsequence).
3. The parts of the *recommendation* that are NOT mapped to the user's timeline and therefore need to be moved from the requirements to the aims of the recommendation (e.g. episodes B and N).
4. The parts of the user's timeline that are NOT mapped to any of the recommendation's timeline. This has no impact on the recommendation process, as it conveys elements of the user's pathway that are not relevant to the current recommendation.

This functionality is still under development and is not yet ready for the upcoming evaluations. It will be described in greater detail in a later deliverable.

5.3 Customisation of the L4All System

The customisation currently works at different levels of the system:

- At the GUI level, where the users can modify the colour scheme used for the visualisation of their timeline and its episodes (see Figure 4)
- At the service level, where previous parameters of a search form are stored and reused for future runs (see Figure 6).

All information is stored in time-limited cookies on the users' browser. Persistence of information across sessions has not been addressed yet, as it will require the extension of the User Model.

6 Conclusion

The development of the first version of the L4*All* personalisation engine has now reached completion and will be used for the first phase of evaluation. Since the recommendation mechanism will require significant work in specifying and implementing a set of actual recommendations, its completion has been postponed to after this first phase of evaluation, in order to focus first on the new interface and the search for "people like me".

This report has covered the following development activities:

- Redesign of the GUI of the L4*All* system, using DHTML/javascript for the front-end and JSP/servlet for the back-end.

- Redesign of several aspects of the ontology underlying the L4*All* system in order to accommodate the new functionalities: different categories of user (learner, expert, institution), a two-axis taxonomy of events, etc.
- Design and implementation of a similarity measure engine for the comparison of learners' timelines. The mechanism is based on converting timelines into strings of comparable tokens and using string metrics for ranking them.
- Design and implementation of a recommendation engine, using the timeline formalism for representing requirements/recommendations and the similarity engine for proposing matches.
- Design and deployment of several customisation procedures (colour/shapes used in the timeline visualisation, bookmarks for interesting timelines, etc.)

The significant effort spent in redesigning the interface comes from the fact that providing a personalised mechanism for searching for "people like me" is only a small – but important – part of supporting lifelong learners in planning their personal development. A crucial aspect of this support is to make sure that learner can *exploit* the information returned by the system. One of the key assumptions for better support is to allow learners to *simultaneously* access several timelines in order to compare them.

The upcoming evaluation will tell us how the design decisions reported in this document have been assessed by two target groups of learners.

7 References

S.E. Ainsworth, D.D. Clarke, R.J. Gaizauskas (2002). *Using edit distance algorithms to compare alternative approaches to ITS authoring*. In S. A. Cerri & G. Gouardères & F. Paraguaçu (Eds.), Proceedings of the 6th International Conference ITS 2002 (LNCS 2363), pp. 873-882. Berlin: Springer-Verlag.

D. Laurillard, A. Poulovassilis, G. Magoulas, S. de Freitas, G. Papamarkos, N. Van Labeke (2007). *WP3: MyPlan Personalisation Specification*. MyPlan Deliverable D3.1 (available at http://www.lkl.ac.uk/research/myplan).

G. Papamarkos, A. Poulovassilis, G. Magoulas (2005). Final Technical Report. L4*All* Deliverable D6.2 (available at http://www.lkl.ac.uk/research/l4all).

8 Appendix: List of similarity metrics

The following list briefly described the similarity metrics used in the document and implemented in the L4*All* system.

- **Levenshtein.** This is the basic edit distance function whereby the distance is given simply as the minimum edit distance which transforms one string into the other. Edit operations are listed as follows:

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

  where d is a function whereby d(i; j) = 0 if i = j, d(i; j) = 1 otherwise.

- **Needleman – Wunsch.** This is similar to the basic Levenshtein distance, by adding a variable adjustment to the cost of a gap, i.e. insertion and deletion, in the distance metric:

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

  where d is an arbitrary distance function between two tokens and G the gap cost function. Note that the Levenshtein distance can simply be seen as the Needleman - Wunsch distance with a gap cost of 1.

- **Jaro.** The Jaro distance metric states that given two strings X and Y , their distance is:

$$Jaro(X, Y) = \frac{1}{3}\left(\frac{m}{|X|} + \frac{m}{|Y|} + \frac{m-t}{m}\right)$$

  where m is the number of matching tokens and t is the number of transposition. Two characters from X and Y respectively, are considered matching only if they are not farther than max(|X|;|Y|)/2-1. Each token of X is compared with its entire matching token in Y. The number of matching (but different) characters divided by two defines the number of transpositions.

- **Matching Coefficient.** The Matching Coefficient is a very simple vector based approach which simply counts the number of terms, (dimensions), on which both vectors are non zero. This can be seen as the vector based count of co-referent terms.

$$Matching(X, Y) = |X \cap Y|$$

- **Euclidean Distance**. This approach again works in vector space similar to the matching coefficient and the dice coefficient, however the similarity measure is not judged from the angle as in cosine rule but rather the direct Euclidean distance between the vector inputs. Below is the standard Euclidean distance formula between vectors X and Y:

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

- **Block Distance**. This is a vector based approach so where 'q' and 'r' are defined in n-

dimensional vector space The L1 or block distance is calculated from summing the edge distances.

$$L_1(X,Y) = \sum_n |X_n - Y_n|$$

- **Jaccard Similarity**. This is another token based vector space similarity measure like the cosine distance and the matching coefficient. Jaccard Similarity uses word sets from the comparison instances to evaluate similarity. The Jaccard similarity penalises a small number of shared entries (as a portion of all non-zero entries) more than the Dice coefficient. Each instance is represented as a Jaccard vector similarity function. The Jaccard between two vectors X and Y is

$$Jaccard(X,Y) = \frac{|X \cap Y|}{|X| \cup |Y|}$$

- **Cosine Similarity**. Cosine similarity is a common vector based similarity measure similar to dice coefficient, whereby the input string is transformed into vector space so that the Euclidean cosine rule can be used to determine similarity.

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

- **Dice Similarity**. Dice coefficient is a term-based similarity measure (0-1) whereby the similarity measure is defined as twice the number of terms common to compared entities divided by the total number of terms in both tested entities. The Coefficient result of 1 indicates identical vectors whereas a result of 0 equals orthogonal vectors.

$$Dice(X,Y) = \frac{2 * |X \cap Y|}{|X| + |Y|}$$

- **Overlap Coefficient**. This is a measure whereby if a set X is a subset of Y or the converse then the similarity coefficient is a full match. Overlap coefficient is defined as:

$$Overlap(X,Y) = \frac{|X \cap Y|}{min\left(|X|, |Y|\right)}$$