The PDP Planning Tool

The TENCompetence WP7 Integrated PDP Planning Tool

Software Documentation

Sandra Gilabert, Davinia Hernández-Leo, Philipp Kärger
SYSTEM OVERVIEW

A demo prototype of the application is at the following URL:
http://upf.ernestoarroyo.com/Planner/.

A demo of the dynamic XML generation is located at
http://planner.ernestoarroyo.com:8080/Planner/MetadataInfo.

System Input

The system input is given by the user profile; it will determine the starting point for exploring units of learning related to the user goals. User interaction will provide implicit feedback and helps the system determine the user interests so that the system might suggest relevant UoL tailored to each learner learning path and goal.

The system might be initialized with a customized learning path listing several subjects, which will be used as a starting point in case there are not initial criteria established. This functionality allows users to navigate and explore the information (knowledge base) without the need of a specific target. The interface then allows learners to discover UoL and related information.

Interface

The user requirements for a planning tool suggest an interface where information exploration is simple and easy to use. With the aim of complying with these guidelines, the interface uses a bubble metaphor to encapsulate a range of subjects. Each bubble (object) represents a unit of learning and contains all information related to the UOL it represents: title, description, prerequisites, etc. Each UoL bubble includes visual information that allows an easy identification. **UoL topic area:** Each bubble has a different color assigned (red, green, blue, etc) depending on the UoL topic area. **UoL size:** Bubble size indicates the number of credits per UoL. These interface mappings are subject change based on user’s feedback and usability evaluations.

The approach of using bubbles as a metaphor to represent information emerged in 2002. *Groxis and Cloudmark* defined a bubble as an information node, which can have other nodes or related bubbles. This metaphor has been selected because of its flexibility, ease of use and ability to represent abstract information, as well as various types of data (text, images, video). The same kind of metaphor is applicable in different contexts while planning learning goals.

UoL description

Each bubble includes a subject description displayed as a “tooltip”. The description summarizes the UoL contents, allowing the users (learners) to quickly grasp the subject gist. The description includes a generic image that is automatically generated based on the UoL short title. These images, together with spatial and visual memory allow for a faster recall.
Filtered visualization

The interface collects information and updates the visual space according to clusters (UoL bubbles) based on the properties identified by the system (provided by the database and API, WP5 services). The interface also displays conceptually related subject topics to the user goals. The figure below shows how the term bubble is used to filter the information presented to the user in order to avoid overloading the interface with visual elements.

![Filtered visualization](image)

Figure 2 – Filtered visualization: reduces information overload

Interface Layout

The interface is divided into two areas: an exploration area with the proposed learning path, and a personalized area where learners can configure their own learning path. Learning paths are organized by a “temporal line” as to provide a path duration cues to the learners. This temporal organization ultimately depends on the learner’s goals.

All units of learning associated with the learning goal float around the interface following a proposed learning path (vertical axis) and organized by relevance to the learning goal.
(horizontal axis) following the ideas illustrated in Figure 3. Units of learning are also linked based on the dependencies among each other.

Figure 3 - Interface: Screen is split into suggested learning path and personalized learning path (proposed path shown)
Figure 4 - Interface: Units of learning are linked based on dependencies among each other.

Figure 5 - Interface: Learners can drag each UoL to the desired position on the user's planning area.
Figure 6 - Interface planning area: Learners can create or redefine a customized learning plan based on their own criteria by organizing each UoL according to their preferences.
INTERACTION DESIGN

This section shows a typical use case of a bubble-based interface for UoLs. The use case is based on the analysis of user requirements. That is also the reason why this section could also help users in their familiarization with the visualization and interactions of the interface.

**Initial visualization: Explore the proposed learning plans**

The user is presented with two interactive areas: suggested learning paths according to the positioning service and planning area for composing the personal learning path. An information area includes all competences associated with the suggested learning path, colored and organized by type.

The suggested learning path visualizes related UoL organized as bubbles with dependency relations. These dependency relations are shown only when the UoLs are selected. The UoLs are also somehow clustered according to the proximity suggested by their content. Their position in terms of the X and Y-Axis is determined by the positioning service. The size of the bubble is the number of credits ("cost" in terms of time) that is estimated to satisfactorily complete the UoL.

Moreover, when a bubble is selected a media visualization associated to the UoL is shown. The media visualization can be text combined with image (and eventually also a video). In case there are several media visualizations associated, they are shown successively. When the bubble is no longer selected, then the media visualization disappears. The bubbles can be redistributed by the user when exploring the area with the suggestions.

**Drag-N-Drop Definition**

Drag-n-Drop allows the movement of objects (UoLs) in the interface. In this case the selected UoLs (from those available in the exploration area) can be moved from the exploration area to the area where the actual learning path is planned (from left to right). Multiple UoLs can be added to the personal plan organized according to a temporal line.

"The user holds down the right mouse button, moves the mouse left, and releases the right mouse button or an alternative way is to hold the right button down and then click the left mouse button."

**Planning and Exploration**

After several iterations (dragging and dropping bubbles, exploring the different UoLs opportunities suggested to achieve the learning goal) users can easily personalize their own learning path. The personalised learning path is organized by time by default and can be easily customized by dragging UoL to the desired position, depending on the learner's needs.
IMPLEMENTATION DETAILS

Initially, the PDP tool was planned to be implemented in Java. A very primitive prototype was developed using this programming language, but during the last few months the implementation requirements changed, and it was agreed that the development should be conducted in Flex. Hence, the expressiveness of the tool could be greatly facilitated. As a first approximation, we have employed Flash to build a prototype that meets the new requirements.

As it has been mentioned in other sections, the planner is aimed to provide users with a highly interactive and expressive representation of the information generated by the positioning and navigation services, which are implemented in Java. Thus, we had to design a system that enables the communication between both programming languages (Java and Flash). We have addressed this problem by using Java servlets, as an interface between the personalization service engine and the Flash web client.

System Overflow

Users enter their personal preferences, and learning goal through the planner. According to those parameters, the client requests a recommended learning plan from the personalization services, which retrieve from a MySQL database the information associated with the input data, and compute the most optimal learning path. For each learning object contained in the plan, its location on the screen is also calculated by the engine. Finally, the planner visualizes the resulting data, which is vertically arranged by time, and horizontally centred by their distance to the user’s learning goal, and personal preferences. At this level, learners are expected to explore, search, compare and select those learning activities that are relevant to their academic goals.

The SQL database currently contains just one competence development program (the Psychology program of the Open University of the Netherlands), plus a default user profile and learning goal. Therefore, the PDP tool just concentrates on requesting the existing data, and provides the user with a highly expressive and interactive visualization of the available information.

System Architecture

As shown in figure 1 (architecture), the Flash Web Client (flash player v8) establishes a connection with a Tomcat Apache server (apache-tomcat-6.0.14), where the servlet is hosted. The learning path plus the description of the learning objects contained in it are requested through a GET message. After this, the servlet invokes a Java framework that communicates the petition to the personalization engine. Once the computation of the learning path is finalized, the personalization engine sends the resulting information to the Java framework, who assigns learning categories and related images (keywords) to the learning activities. Finally, the servlet collects a Java object that encapsulates the result of the whole process, and traduces it into an XML file, which is sent back to the Web Client. In figure 2 and 3 show an example of the XML files describing the learning path, and learning objects metadata.

The image attribute of the metadata XML file, is used by the web client as a keyword to retrieve an image from a database, which is a visual description of the learning object. On the
other hand, the category field groups learning activities that are semantically related. The web client displays the categories in different colours.

**Image retrieval**

200,000 Images were extracted and their labels clustered from istock photo. The clusters represent the images that best describe each unit of learning. A image service running on the server returns an image based on a given keyword. This generic approach to image retrieval allows the interface to grow without manual intervention as more competence development program become available.

The image attribute of the metadata XML file, is used by the web client as a keyword to retrieve an image from a database, which is a visual description of the learning object. On the other hand, the category field groups learning activities that are semantically related. The web client displays the categories in different colours.

![System Architecture Diagram](image)

Figure 1 – System Architecture
De biologie van het gedrag

De biologische psychologie is de wetenschappelijke bestudering van de biologische basis van het gedrag en mentale processen van mensen. Echter, ook de resultaten van dieronderzoek (dat strikt genomen onderdeel vormt van de neurowetenschap) vormt een bela

De persoonlijkheid is het geheel aan eigenschappen die men aan een levend wezen kan verbinden. Het geheel van deze eigenschappen vormt het wezen zijn persoonlijkheid, die men kan karakteriseren in bepaalde termen die hanteerbaar zijn op dat vlak. Zo kan

The Individual

Social Psychology Foundations

The Individual

Human behavior

child
Client side

The Flash client is responsible for showing the information received from the server in a convenient way, and provides the interactivity needed for the use of the application by the user.

The interface performs a petition to the server requesting the XML document of a given learning path. The returned XML, as mentioned before (fig.2 and 3), includes the general structure of the studies as well as the detailed information of every unit of learning.

In the interface, a unit of learning is mapped to an ActionScript 2.0 Node class that is linked to a graphical object, which is rendered as a circle containing the short name of the learning activity. This object contains all the information related to the learning object, and enables the interaction between the user and the unit of learning (it answers to events like click, hover, or drag). The graphical characteristics of the circle are parameterised by some of the attributes of the learning objects. For instance, the dimensions of the circle are given by the cost of the activity, whereas colour depends on its category.
The interface creates internally two node groups: one for the proposed plan, and another for the user one. All the learning objects retrieved from the XML are added to both groups. However, all the nodes in the learner plan area are initially inactive (not visible). When the user drags a unit from the proposed plan (left area) to the learner plan (right area), the dragged node is activated in this area, and hided from the other.

Each node group keeps a link list between the different units of learning. The dependency relations among objects are mapped to an ActionScript 2.0 graphical object (an arrow).

All this data is managed by another ActionScript class, the Manager class, which establishes the connection with the Tomcat server, requests the data to be displayed, and controls user’s interaction inputs (dragging, hovering, …).