Computational Instructional Design for Construction of Adaptive Tutors in Real Time from Distributed Learning Objects

Kurt Rowley

Command Technologies, Texas, USA

With the advent of the WWW and recent interoperability and ontology initiatives, there are new possible approaches to generating real-time adaptive instruction by drawing from distributed interoperable learning object resources. In order to assemble adaptive tutoring systems appropriate to the needs of individual students, a computational form of instructional design has been developed with a generic instructional vocabulary. The vocabulary facilitates the automatic construction and management of tutoring environments from distributed interoperable components. A demonstration web-based ITS that generates individualized and adaptive instruction in English Composition through the use of an interoperable instructional vocabulary is used for illustration.

Influence of Didactic and Computational Constraints on ILE Design

Nicolas van Labeke[1], Josette Morinet-Lambert[2]

[1]Universite Henri Poincare – Nancy, France, [2]Vandoeuvre les Nancy, France

Our major research aim is to analyse the teachers' needs and to propose a design process taking into account teachers as users. The main characteristics of this project are: 1) the development of *Calques 3D*, a microworld for spatial geometry learning, 2) a pluri-disciplinary work, 3) a framework to lead the negotiation between the different actors, and 4) a production cycle based on a step-by-step upgrading of a prototype. The ILE design has underlined the need of negotiation between the involved partners, since it may result in didactic or computational transpositions of the teaching domain. On the one hand, teachers have often to adapt the domain they teach (e.g. filter, simplification, metaphor, etc.). On the other hand, the software engineers have to restrict the required implementation due to development constraints or ergonomic requirements. This work allows us to focus on three *design rules* induced by the need for maintaining teachers' agreement. First, it appears that any pedagogical choices, even accepted during the design process, could be mistrusted by user-teachers. It implies that an adequate set of *environments parameters* should be implemented in order to allow the teacher to adapt the software according to his usage. Secondly, we consider that each constraint or property added to the teaching domain by the software engineers, for computational reasons, have to be expressed explicitly (e.g. by enhancing graphical interface rules). Finally, adding software features requested by teachers could increase the environment complexity and thus reduce its conviviality. So it appears essential to simplify user-interface either permanently or during a particular task (e.g. by curbing ILEs).

B.P. Goettl et al. (Eds.): ITS '98, LNCS 1452, p. 615, 1998.