



The ExploraGraph Advising System : an Ergonomical Evaluation of the Editor

Aude Dufresne, Céline Schlienger

► To cite this version:

Aude Dufresne, Céline Schlienger. The ExploraGraph Advising System : an Ergonomical Evaluation of the Editor. Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie, Nov 2002, Villeurbanne, France. pp.299-305. edutice-00000668

HAL Id: edutice-00000668

<https://edutice.hal.science/edutice-00000668>

Submitted on 19 Oct 2004

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

The ExploraGraph Advising System : an Ergonomical Evaluation of the Editor

Dufresne, Aude , Céline Schlienger**

Université de Montreal, C.P. 6128, Succ. Centre-Ville, Montréal, QC, Canada H3C 3J7

Associate researcher, LICEF – dufresne@com.umontreal.ca

Abstract

The first aim of the ExploraGraph project was to design a better interface for learners taking distance courses and also to integrate in it adaptive functions and advices to support them. The system offers a generic interface, in which courses can be defined as conceptual graphs describing structures of concepts, activities and/or documents. ExploraGraph serves as a front to access web courses.; local and distant applications, documents and forums may also be organized into a course. The system uses a Navigator for learners and an Editor for professors to design the learning and supporting environments. We present here the results of the ergonomical evaluation of the ExploraGraph Editor. Observation of professors trying to use the system to describe a course highlight the complexity of the activity of creating courses as a structure of activities integrating multiple references, but especially to design support functions in such a learning environment, taking into account individual progression. This research presents how the system could be made more generic, using templates of generic types of support. Thus we propose elements for an ontology of support in telelearning environment

Keywords : telelearning, interface, ergonomy, evaluation, support system, ITS, knowledge engineering, ontology.

Résumé

L'objectif du projet ExploraGraph était de concevoir des interfaces adaptées pour supporter les activités de l'apprenant suivant des cours à distance, qui intègre aussi des fonctions adaptatives de soutien et de conseil. Le système offre une interface générique, qui permet au professeur de définir l'accès à un cours au sein de graphes conceptuels présentant les structures d'activités, de concepts et/ou de documents. ExploraGraph sert ainsi d'interface pour accéder à des documents ou à des applications locales ou sur Internet. Le système est composé d'un Navigateur pour les étudiants et d'un Éditeur pour les professeurs. Nous présentons ici les résultats de l'évaluation ergonomique de l'Éditeur ExploraGraph. L'observation de professeurs apprenant à utiliser le système pour définir des cours fait ressortir la complexité des activités de création d'un cours intégrant de multiples références, mais en particulier d'associer des fonctions de soutien, qui tiennent compte de la progression individuelle. La recherche suggère des modifications visant à rendre le système plus générique en utilisant des gabarits de règles génériques. Nous proposons quelques éléments d'une ontologie du support au sein d'un environnement de téléapprentissage.

Mots-clés : téléapprentissage, interface, évaluation ergonomie, tutoriel intelligent, acquisition des connaissances, ontologie.

Context of the Research

The ExploraGraph© system was developed in collaboration with the LICEF, based on their distance learning architecture model (Paquette, et al. 1996). It was part of the Human Interface project in the context of the Canadian TeleLearning Network of Centres of Excellence.

This research (Dufresne, 2001a) stems from a preoccupation for interface and communication design in ITS especially in the context of distance education, in very open domain of learning. Many ITS models depend on a highly detailed representation of the domain to be learned, where knowledge can be tested at micro level and inference can be made among knowledge elements. It is then possible to design natural language understanding or specific explanations or deitic demonstration for a given problem or scene layout. Our problem was to generalize the principles of support, to make them accessible when the models of the domain were more shallow; when the programming of support was to be done by professor with no special training. In such context, evaluations cannot be as detailed and more self-assessment is necessary to access the learner's model.

The support functions were to be integrated in a general course editor and have to be based on principles that could be applied to extract constraints in tasks and to give advice; to present demonstration and suggest content elements using a generic editor. In other to enrich the contextual model and also the means of support we developed a dynamic and adaptive interface in which we experimented various dimensions of personalized support.

The first aim of the ExploraGraph project was to design a better interface for learners and to integrate in it support based on an "overlay model" of the learner. The interface was designed to make conceptual and pedagogical structures visible, and to use this interface as the gateway to the course. The access through the conceptual structure front-end, made it possible to keep a trace of the learner activity and to use it to give him feedback on what he had done, what he had to do. The learner could control his learner's model, by marking what he thought was finished as he progressed. A structure of possible goals or intentions were also

accessible to the learner, so he could choose an intention and the interface would advise him and support is exploration, by popping the right graph, the right description of activity depending on where that specific learner was in his progression. In the last version of the system, the learner could even express preferences for help (more or less) and choose a coach with a specific personality. Coaches were different Microsoft agents avatars, each one addressing a specific type of learner (Martinez, & Bunderson, 2000). Different rules for support could be defined depending on the context – task, progression, intention, preferences and coaches (Dufresne, 2001).

Though the principles of the interface were promising, the bottleneck of the system was the definition of the rule-based support system. The more we were refining the model and adding complexity to it, the more intractable became the possible combinations of rules and the dynamic interaction among them. In fact the ExploraGraph system was a designed out of human-computer interaction preoccupation to improve the interface for the learner, but the same approach was now needed to facilitate the task for the author of the ITS. We studied how the

Editor was being used, in order to study the course editing process and to improve the interface to support this task (Schlienger, 2001).

Ergonomical Evaluation

The ExploraGraph Editor was developed initially as a mean to describe the conceptual graphs structures of the Navigator so the learner interface could be tested. The rule definition interface was later added, so different hypothesis on the structure of the support system could be tested (figure 1). In this context the interface of the conceptual graph Editor and of the advising system had only been used by the researchers themselves. They had gained experience on developing and debugging the conceptual graphs and the help system, using some testing and some experimentation with learners, followed by inspection of the traces of execution. If the first prototype of the Editor was found useful to make the proof of the concept of this form of adaptive and contextual support, but many limitations were found in the validity of the developed applications:

The screenshot displays the ExploraGraph Editor interface, which is divided into two main windows: 'Gestionnaire d'événements' (Event Manager) and 'Action'.

Gestionnaire d'événements (Event Manager):

- Nom de l'aide:** DessinerN
- Conditions:**
 - Contexte:** Masse clic
 - Valeur1:** (empty)
 - Valeur2:** (empty)
 - Nom du graph:** Flash5Animations
 - Nom de la masse:** Dessiner
- Trace:**
 - Répéter l'aide:** 7 fois
 - Nom action:** (empty)
 - Options:** Oui (selected), Non
 - Nom aide recue:** (empty)
 - Options:** Oui (selected), Non
- Enregistrement:**
 - Complétée (%):** (empty)
 - Visitée (nombre):** (empty)
 - Délais Session (Jour):** 0
 - Durée Session (hrs):** 0
 - Enregistrer:** (button)
- Actions:**

IdAction	NomAction	NomContrô
110976	DessinerN1	MSAgent->
5213854	HackerDisparait	MSAgent->

Action Window:

- Nom de l'action:** DessinerN1
- Nom du contrôle:** MSAgent
- Identifiant du texte:** 1352360517
- Nom de la méthode:** Speak
- Fichier Son:** ouverture12.wav
- Agent:**
 - Choisissez un agent:** Hacker
 - Lier un geste:** (button)
 - Lier un son:** (button)
- Texte de l'agent:** Je veux juste vous rappeler que cette partie-là, vous l'avez déjà fait il y a deux semaines, alors perdez pas trop de temps dessus.
- Ajouter:** (button)

Figure 1. Interface of the ExploraGraph Editor where the **conditions** of appearance of a rule and the support **actions** are defined.

It was difficult to create the conceptual graphs and to assess their completion and validity (missing links, misuse of the conceptual ontology, incomplete definitions). Even for us, it was almost impossible to create a valid rule-based support system, insuring its validity and completeness – misplaced, incomplete and overlapping support rules.

The general model of support was still lacking a more generic model that could guide the development of the rules.

But more importantly we wanted to test the system with the intended users (pedagogical engineers), so we could make it usable for them. So we found it important to make a more thorough analysis of the Editor Interface with real users, in order to test and improve it. Nielsen (2001)¹ said ‘usability testing means more than putting products through their paces in a lab’, it has to be experimented on real tasks with the targeted users.

Methodology

Observational studies are a concrete mean of analysing the *real* user interacting with the *real* system and not just for checking usability issues per se but also how a system is useful to the task when one uses it. This evaluation method means that the user is involved in the software design process right from where and for what they need it. As far as ExploraGraph was concerned, we complemented user testing with a focus group and interviews as well. Also, to test for the suggestions we used a low-fidelity prototype.

ExploraGraph Editor empirical evaluation targeted two goals: 1) to test the software usability in itself and 2) to gain suggestions on some extra functionalities to supplement it. Ease of use, ease of learn, usefulness of the features offered and wording were the principal issues that we investigated. The goal of the study was to simulate the real use of ExploraGraph while completing predefined tasks.

During a test (see figure 2), each subject was hooked up to a microphone to record what he said. Each user was accompanied by a person responsible of the experimentation next to him and a third person placed behind them to take notes of what was going on between them and the machine. The latter was asked to think aloud all impressions, questions and suggestions that could possibly run through their mind. Although the verbal protocol has its drawbacks (i.e. interrupting the user while he completes his task), it certainly gives access to spontaneous feedback from the user gathered *in situ*. Afterwards, people tend to forget the problems they encountered using the system.

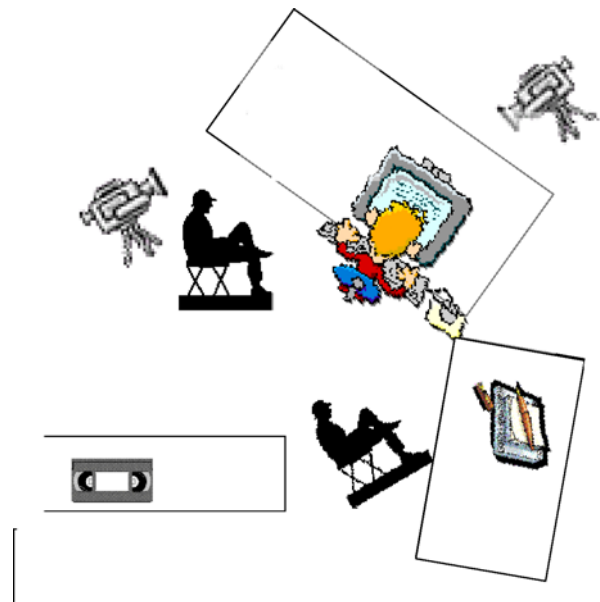


Figure 2. Experiment of the ExploraGraph Editor.

To explore how the ExploraGraph could be improved, we prepared a paper prototype, which was used after the real system to get feedback on potential modifications of the system. We asked the users to imagine how they would define contextual help with both ExploraGraph and the paper prototype, so that we could observe the pros and cons of the each version. At the end of the experiment, users were invited to comment on their experience and to share their general feelings and thoughts about the software.

For each major functionality, we started out breaking down the tasks in sub-tasks to cover the various ways possible to complete a single task. We then invented a scenario to put the user in a make-believe situation where they were asked to accomplish some work which involved to create a new graph, to draw it with its nodes and links related to some content and to define contextual help. Small texts were given as guidelines as to what help was necessary and for what purpose.

As a tool to collect the data more easily, an observation grid was developed taking into account:

- the time to complete the task,
- the ExploraGraph help index chapters consulted if any,
- the questions asked by the subject,
- the mistakes made by the subject,
- the subject's satisfaction where applicable,
- the user's actions
- and a last column was devoted to questions to ask the user during the test, to investigate problems.

Each testing session lasted three hours. After greeting the subject and thanking him for participating in the test, we proceeded with describing the test agenda. A questionnaire was then handed out while we were reassuring him that all data collected would remain anonymous and confidential. ExploraGraph

¹ “Ease of use doesn't come easy”. Electronic business, 12/01/01. <http://www.e-insite.net/eb-mag/index.asp?layout=article&stt=000&articleid=CA183275&pubdate=12/1/2001>

was then presented to the tester and he could even try the software out. We would show her what the application was for and hence allowed her to see the result of what would be asked from her, reminding her of course that the machine was the one tested through the experiment not the person... After insisting on the importance of the think aloud process, the subject was left to go through the prepared scenarios in a maximum of two hours.

Subjects

Before starting the empirical study, we chose the subjects very carefully. On line or distance course designers were the principal audience targeted since The ExploraGraph editor is aimed at helping them define support for their course. A brief questionnaire was to be filled by the chosen testers so as to collect information that would help interpret results afterwards if necessary.

For the experiment ten (10) participants were chosen among graduate students, research professionals and professors but all involved in distance education. All but a few had experience designing telelearning

courses and none had taken any as students. Some had already seen demos of the ExploraGraph system. It is important to mention here that to make sure the tests would run smoothly, we had run some pre-tests beforehand.

All the test sessions took place within the timeframe of a week. Although the test as a whole ran for three hours, the subjects remained on task and even mentioned their appreciation of the experiment at the end. All subjects succeeded in creating graphs for a course, while they almost all failed at defining help. Some lacked time while others had barely the time to define some which allowed only four participants to test the low-fidelity prototypes.

Results

Overall, participants liked the way ExploraGraph allowed to visually navigate through a course content with graphs and the idea of being able to define contextual help themselves through an editor appealed greatly to them, even though they all agreed in saying that it was too complex to use in its actual state.

Erreur ! Liaison incorrecte.

Table 1 Structure of the support system

The overall analysis of the observations lead to the description of four different types of usability problems:

- the way information was presented - Too many parameters to set, lacking organization, feedback and affordance². There were too many levels in the definition of the support structure :
 - association to a course element;
 - contextual triggering event and conditions
 - support defined as group of actions
 - individual actions with their parameters.
- the wording used, especially in relation to the physical description of the elements in the animation (mass, elasticity of links), or in the description of possible events, conditions or support actions in the environment.
- the modality of dialog chosen (For example, the node creation was only possible using keyboard-based commands. It was not possible to create graphs, using the menus or a toolbar)
- the understanding of the software functionalities and architecture (ExploraGraph features are unusual and complex, like for instance the fact that a node was an instance of a content but was

not the content itself was not easy to grasp at first).

The subjects made numerous suggestions during and after the experiment. This corresponds to the three categories of the Seeheim's User Interface model (Green, 1985) :

- Presentation lay out (what the user sees)
- Dialog (how the user communicates with the system)
- Functional core or kernel (model of data processing behind the interface).

As for **presentation**, it was difficult to organize all the parameters that could possibly define the support in the ITS (see Table 1). We found that the hierarchy was difficult to express and should be scaffolded in the visualization.

In fact there was no view of the general structure of decision, **no model of the ITS functional core processing** that could have helped the conceptors understand the organization in the editor interface or could have made help them follow the execution of the program to verify it.

Since most users had no training in knowledge engineering, they were completely lost when asked to define the support system. For them the planning and validation of the rules were far too complex.

Not only were the rules difficult to define, but following the interactions among rules, was intractable. For example they were relatively few events that could be used to trigger a support rule.

² « Ensure an object displays good affordance. That is, the user can easily determine the action to be taken with the object. » IBM ® User Interface Architecture, Copyright IBM Corp. 2001

opening of a graph;
 clicking on a node;
 choosing an intention;
 idle time
 date
 marking an activity as completed.

We had planned to present a calendar (see figure 3)
 to view the organization and extension of rules in time
 and access them classified by event. But it was
 insufficient when many rules were overlapping for a
 period, more sorting and filtering should be offered to
 insure visibility for design.

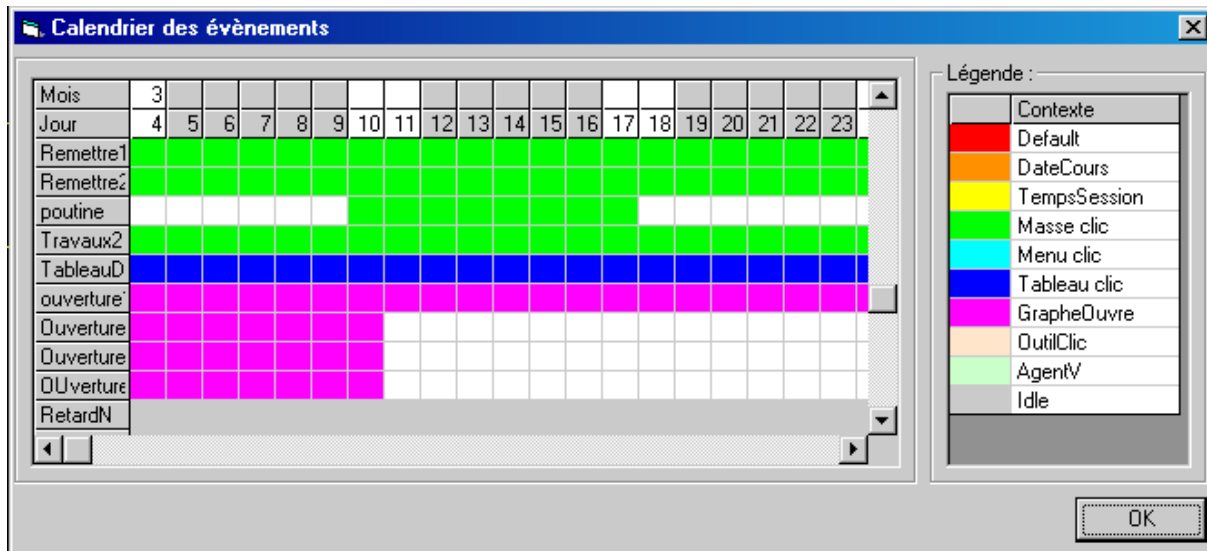


Figure 3. Calendar showing the rules classified by events (color coded).

Also the definition of rules for each possible preferences were hard to instantiate in reality. Though the system was made to search and order rules according to elements of the course; it was almost impossible to know clearly where rules were missing, and how they would be triggered in practice.

Discussion and conclusion

Toward a more usable knowledge engineering tool for ITS.

The questions that were posed by the empirical evaluation of the ExploraGraph Editor, where far more complex than modifications in the lexical choices. In fact, it suggested to restructure the rule-based system, so it would be organized at more understandable level for authors. There was no way the support could be described starting at the level of rules, it was necessary to start at a higher level, at least to give a schema of the model of support that could be understandable.

In order to improve tools to define ITS, one must refer to general references on knowledge acquisition (Breuker, & Wielinga, 1987; Clancey, 1983; Dufresne, et al. 1992; Nwana, et al. 1991; Sowa, 1984) and to offer a representation and tools where the general organization is highlighted. As Chandrasekaran (Chandrasekaran, 1985) proposed when defining knowledge based system, "it is important to use the right level of abstraction, to structure expertise and make it understandable". Thus in traditional knowledge

engineering approach, general models of decision can be used to describe decision at a higher level of abstraction before they are applied to specific cases. In the domain of ITS different such higher level models have been proposed.

In fact the problem of defining an ITS is solved differently depending on the type of ITS. For example, for computer simulation laboratories, the tutoring is linked to the expression of the principles behind the simulation. Ideally, an interface manager for structuring the interaction and support should be separated from the simulation model, but most often help is simply embedded as general or contextually accessible help, but which don't adapt to the learner understanding or progression on the task. Very often, the tutoring associated with a specific simulation is left to a teacher who coach students on using the system.

(Lajoie, Faremo, & Wiseman, 2001) proposed principles that can be used to define case based ITS, showing how cognitive apprenticeship theories can help design support to guide the learner understanding of the cases at hand : develop generic competencies, perform steps in diagnosis, applying principles to cases. Using the verbal protocol data of experts giving support, they propose different generic types of interventions, like bridging between the cases and between cases and the metalevel of explanation.

Other authors suggest ontologies based on pedagogical models to define support at a metalevel.

(Guin et al.,1993) suggests to structure the support interaction using agents with different strategies:

- the *oracle* - to declare knowledge;
- the *questioner* – to question knowledge in order to test and draw attention to difficult elements;
- the *technicien* – to describe technical procedures, to comment and to give examples;
- the *master* – to judge and evaluate the progression.

(Frasson, Mengele, & Aimeur, 1997) also suggest to have various actors interacting to suggest support; they propose the role of the *trouble maker* as another interesting metastrategy to provoke the learner in his understanding.

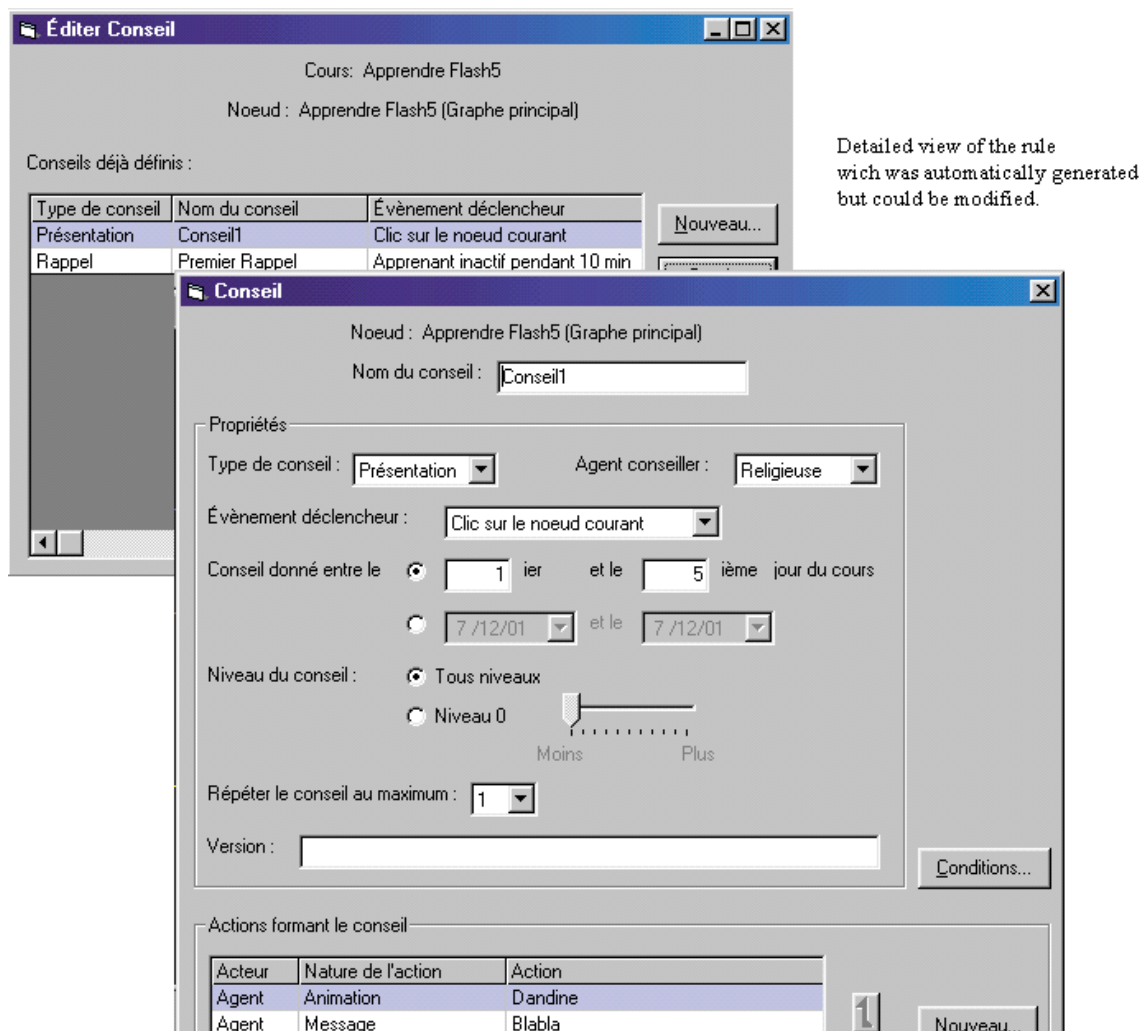


Figure 4. Prototype of a more generic interface for rule editing, based on templates of pedagogical support rules – presentation, reminder, collaboration suggestion, etc.

Using a metalevel representation to define rules

In the context of ExploraGraph, the support may be linked to the structure of the course, to the conceptual graphs, to the learner's choice of intentions and to the menu of the application. Aside from this structure of the application, a pedagogical structure of support goals can be used as a mean to define templates to the rule based system.

Figure 4 presents a proposition for a new version of the system relying on such metalevel of strategies. For example, choosing an activity, the author would associate an **Explain** and a **Reminder** support rules.

The choice of a template would then instantiate a typical conditions and support actions associated with the templates.

- Present – help accessible at login in the week of an activity, where the system open the description of the activity, with a general comment of presentation.
- Explain – detailed description accessible when the right button is pressed, or when the learner choose the intention –“Explore the activities” in the week of the activity.
- Reminder – help accessible 5 days before a deadline, when the activity is not marked as completed.

The rules defined using templates could then be modified to adapt them to specific cases: change de default message, change the timing of the reminder, etc.

The structure of the types of support actions was also reorganized, whether controlling the application or displaying messages. For support actions also, the system would set by itself some of the parameters (default avatars, default node to activate, etc.). In fact even the structure of the control system of supports should be redesigned to support more goal directed organization of support, where each agents collaborate in a more competitive and complementary way to define the best support at on point in the learner progression; each contributing to diagnosis and to define parts of support messages.

- try to motivate when the learner is not performing as he should using non-verbal expressive behavior.
- support every work session, only once;

Aknowlegments

This research was granted as part of the Human Interface project in the context of the Canadian TeleLearning Network of Centres of Excellence.

References

- Breuker, J., & Wielinga, B. (1987). Knowledge Acquisition for Expert Systems: A Practical Handbook. In A. L. Kidd (Ed.), *Knowledge Acquisition for Expert Systems: A Practical Handbook* New York: Plenum Press.
- Chandrasekaran, B. (1985). *Generic tasks in Knowledge-based reasoning: characterizing and designing expert systems at the "right" level of abstraction*. Second Conference on Artificial Intelligence Applications: The Engineering of Knowledge-Based Systems, Miami Beach(pp. 294-300).
- Clancey, W. J. (1983). The Epistemology of Rule Based System -a Framework for Explanation. *Artificial Intelligence*, 20, 215-251.
- Dufresne, A. (2001a). Conception d'une interface adaptée aux activités de l'éducation à distance - ExploraGraph. *Sciences et Techniques Éducatives*, 8(3), 301-320.
- Dufresne, A. (2001b). ExploraGraph : Improving interfaces to improve adaptive support. In J. D. Moore, C. L. Redfields, & W. L. Johnson (Ed.), *AIED'2001* (pp. 306-313). Amsterdam: IOS Press.
- Dufresne, A., Muzard, J., Legault, B., & Dufour, S. (1992). ALADIN: Une interface pour supporter le développement des bases de connaissances. *ICO*, 4(1-2), 37-46.
- Frasson, C., Mengele, T., & Aimeur, E. (1997). *Using Pedagogical Agents In a Multi-strategic Intelligent Tutoring System*. <http://citeseer.nj.nec.com/frasson97using.html>.
- Green, M. (1985). Report on Dialogue Specification Tools. In G. E. Pfaff (Ed.), *User Interface Management Systems* Berlin: Springer Verlag.
- Guin, D., Billet-Coat, S., Rietz, P., & Hérim-Aime, D. (1993). Protocole comportemental de l'interaction didactique entre un agent artificiel et un agent humain. In M. Baron, R. Gras, & J.-F. Nicaud (Ed.), *Environnements interactifs d'Apprentissage avec Ordinateur* (pp. 193-205). Paris: Eyrolles.
- Lajoie, S., Faremo, S., & Wiseman, J. (2001). A Knowledge-Based Approach to Designing Authoring Tools : From Tutor to Author. Amsterdam: IOS Press, 306-313.
- Martinez, M., & Bunderson, C. V. (2000). Foundations for Personalized Web Learning Environments. *ALN Magazine*, 4(2), http://www.aln.org/alnweb/magazine/Vol4_issue2/bunderson.htm
- Nwana, H. S., Paton, R. C., Bench-Capon, T. J. M., & Shave, M. J. R. (1991). Facilitating the development of knowledge based systems : a critical review of acquisition tools and techniques. Avignon 91, Avignon. (pp. 487-500).
- Paquette, G., Ricciardi-Rigault, C., Paquin, C., Liégeois, S., & Bleicher, E. (1996). *Developing the Virtual Campus Environment*. ED-Media International Conference, Boston.
- Schlienger, C. (2001). Maquettage et spécifications d'une interface graphique d'édition de cours (projet ExploraGraph). LICEF.
- Sowa, J. F. (1984). Conceptual Structures, information processing mind and machine. Addison-Wesley.
- give positive feedback when something is new and positive in the learner's model
 - present to the learner every new participant, not more than one every X minutes.
 - Give feedback on the group model, every time the discrepancy between group and individual models are greater than 20%
- To implement this more generic goal structure in the support system, generic templates for rules should be designed that can be instantiated and adapted to specific cases.