Providing adaptation and guidance for design learning by problem solving: The design planning approach in DomoSim-TPC environment

Miguel Á. Redondo a,*, Crescencio Bravo a, Manuel Ortega a, M. Felisa Verdejo b

a Universidad de Castilla-La Mancha, Departamento de Informática ESI, Paseo de la Universidad, 4 – 13071 Ciudad Real, Spain
b UNED, Departamento de Lenguajes y Sistemas Informáticos ETSII, Ciudad Universitaria, s/n – 28040 Madrid, Spain

Received 4 December 2004; accepted 21 April 2005

Abstract

Experimental learning environments based on simulation usually require monitoring and adaptation to the actions the users carry out. Some systems provide this functionality, but they do so in a way which is static or cannot be applied to problem solving tasks. In response to this problem, we propose a method based on the use of intermediate languages to provide adaptation in design learning scenarios. Although we use some approaches which are familiar from other domains (e.g., programming tutors) they are novel as regards their application to a very different domain and as a result we have incorporated new strategies. The purpose of our proposal is to provide monitoring, guidance and adaptive features for PlanEdit, a tool for the learning of integral automation methods in buildings and housing by design. This tool is part of a collaborative environment, called DomoSim-TPC, which supports distance learning of domotical design. We have carried out an experiment to obtain some data which confirm that our position can be effective for group learning of domotical design, studying the relationship between the quantity of model work carried out and the errors made.

© 2005 Published by Elsevier Ltd.
1. Introduction

Design and simulation environments for learning should provide students with mechanisms to enhance effective learning. It is therefore useful to interactively monitor the learners while they are solving problems (Verdejo, 1992). With such a functionality, the system can analyse and propose actions taking into account the state of the learners’ work. Furthermore, the system can give advice regarding the instructional level of the problem best suited to the students’ knowledge level.

In order to monitor the student’s work the system needs an explicit representation of domain knowledge as well as of the student’s behaviour. The representation and acquisition of this information has been a primary concern in the field of Adaptive Hypermedia Systems (Trigg & Weiser, 1986) and particularly in the area of Intelligent Tutoring Systems (Murray, 1999).

At present, there is a significant number of studies dealing with the creation and organization of adaptive hypermedia contents and on the design of their browsing. These studies are representative examples illustrating techniques for providing adaptation. The current tendency is to use them on the Web. In these studies, several methods of adaptation have been proposed according to a variety of parameters. A classification of the most frequently used criteria for adaptation is described by Gutiérrez and Pérez (2001). Among them the observation of the user’s behaviour stands out as the most typical approach. As examples of these systems we can mention HyperTutor (whose lines have been commercially reflected in HEZINET (Pérez, López, Gutiérrez, & González, 2000), Metadoc (Boyle & Encarnación, 1994), Hypercase (Micarelli & Sciarrone, 1996), Hyperflex (Kaplan, Fenwick, & Chen, 1993) and ITEM/PG (Brusilovsky & Zyryanov, 1993).

Tools like AHA (De Bra & Calvi, 1998) and TANGOW (Carro, Pulido, & Rodríguez, 2000) allow the presentation of the information published on the Web to be adapted. They do so according to the users’ characteristics (profiles). They use conditional sentences and other rules that determine the contents to be shown and the structure of the course. This adaptation process is carried out in an eminently static way and takes place when the tasks of the course are proposed.

Other systems such as DCG (Vassileva, 1997) and ELM-ART (Weber & Specht, 1997) take into account the knowledge acquired by the user and dynamically propose the best path to browse the contents of the domain.

According to Jonassen (Jonassen, 2000), it is difficult to use these systems when the aim is to find the solution to a problem. As a result, we propose a method based on the use of intermediate languages to provide adaptation in design learning scenarios. The purpose of our proposal is to provide monitoring, guidance and adaptive features for PlanEdit, a tool for the learning of integral automation methods in buildings and housing by design. This tool is part of a collaborative environment providing support for distance learning of domotical design.

This paper describes our method to provide monitoring, adaptability and guidance features in PlanEdit. Thus, a relevant description of the implementation of automated adaptive tutoring and its evaluation is made. PlanEdit is an evolution of the planner of the DomoSim environment (Bravo, Ortega, & Verdejo, 1999). PlanEdit supports collaborative tasks, while Domosim is non-collaborative. The term Domotics refers to the set of elements that, installed, interconnected and
automatically controlled at home, releases the user from the routine of intervening in everyday actions and, at the same time, provides optimized control over comfort, energy consumption, security and communications.

The DomoSim environment is based on Soloway’s position related to the use of intermediate languages in order to help the student in the resolution of complex problems using the structuring that these languages provide (Soloway, 1986). The environment includes a knowledge-based editor for planning the design model whilst satisfying a set of requirements. In addition, the finished model can be simulated in order to test its behaviour. The editor is able to guide the students’ work, based on general schemas of resolution (design plans or design strategies), and to offer help to solve a proposed problem. However, DomoSim lacks customization capabilities and dynamic adaptation to the users’ actions and does not support collaborative activities of domotical design learning. In some of the experiments with the DomoSim system we soon detected the need to give support for collaboration. With this in mind, we have developed a new environment based on the same theoretical foundations of individual learning but offering support for activities of collaborative learning and new functionalities, mainly for adaptation, monitoring, guidance and simulation. This new environment is called DomoSim-TPC. The abbreviations we have added (TPC) refer to the new functionalities that have been incorporated. PlanEdit is one of the tools that has been added to this new environment. In fact, PlanEdit can be considered to be the evolution of DomoSim, since it offers the same functionality whilst incorporating new adaptation and guidance characteristics which are described in this paper. PlanEdit allows learners to interactively define a plan to solve a modelling problem whilst satisfying a set of requirements. The tool adapts to the characteristics of the problem to be solved and to the strategy that the students adopt. For this purpose, an analysis based on the expert knowledge of the domain and on the user behaviour guides the user to reach the best possible solution.

Using PlanEdit and the DomoSim-TPC environment we have carried out some experiments in Secondary and Higher Education. For example, we carried out an experiment with users in which our aim was to obtain some data confirming the effectiveness of our proposal studying some significant variables. In this experiment, we determine if the students can learn specific aspects of the domain and techniques of group work.

The paper is organized as follows: in Section 2 the background for the targeted learning situations is described and in Section 3 the characteristics of our method and the tool we have developed are discussed in depth; Section 4, looks at some details relating to the technology used for implementation; Section 5 describes an experiment with students; finally in Section 6, we draw some conclusions and outline the future work we have planned to develop.

2. Teaching design procedures in Domotics

As we have pointed out previously, the domain where our research is applied is the learning of the design of automated control installations in buildings and housing. In Spain the regulation for “Formación Profesional” (Technical Training) takes professional profiles into account and training in Domotics is considered to be very important. Some learning stages in electricity and electronics courses focus on the study of the design and maintenance of singular installations and
automation of buildings dedicated to housing. In this area the design of domotical installations has a fundamental role.

In this kind of training, the performing of practical experiments is particularly important. However, the material necessary to carry out these assignments is usually expensive and in many cases it is not adequately provided. This problem is aggravated by the difficulty encountered in bringing the student closer to real situations, in replicating accidents and in simulating those chaotic situations which may arise in the real world. In particular, the domotical installations should be able to provide an appropriate response to these situations.

In order to alleviate this problem by means of the use of technology, we have developed a distributed environment with support for distance learning of domotical design: DomoSim-TPC (Bravo, Redondo, Bravo, & Ortega, 2000). It includes configuration, performance, tracking, tasks analysis and storage of collaborative learning activities. PlanEdit is one of the tools included in this environment and it approaches learning from a personal perspective. Nevertheless, this tool is used as part of a task in which students organized in a group build a design solution to a proposed problem.

With the use of DomoSim-TPC, the teaching of this subject in Secondary and Higher Education also involves a modification of the educational protocol, while continuing to be based on the paradigm of the Problem Based Learning (PBL) (Savery & Duffy, 1996) and its application to collaborative environments (Koschmann, Kelson, Feltovich, & Barrows, 1996). First, the teacher carries out a presentation of basic theoretical contents. Next, students are organized in small groups to whom the teacher assigns the resolution of design problems selected from an available library of problems (Redondo, Bravo, Bravo, & Ortega, 2003a). The characteristics of a problem are described by attributes, including specific didactic objectives. A problem can be selected according to the teacher’s approach. This information is used to induce the adaptation of a general resolution schema for a kind of problem. This adaptation is carried out according to the needs, restrictions and help level established by the teacher. The students use PlanEdit to plan the design strategy in order to build a model they consider will satisfy the requirements of the proposed problem. This plan will be refined in a model at a later stage. Later on they can discuss their proposals. The system offers facilities to collaboratively comment and justify the design decisions taken. Additionally, they can carry out a simulation of a model to check its behaviour under extreme circumstances; and by so doing, they can test if their solution fulfills the requirements (Bravo, Redondo, Ortega, & Verdejo, in press).

3. PlanEdit

PlanEdit is a tool to support the design of models that should satisfy a specification. The design is approached in two steps: first a plan is built and then the plan is developed. A plan is specified as a partially ordered set of generic actions for the construction of a model. PlanEdit helps the users to build a plan. To accomplish this, it uses knowledge about the students and pedagogic information of the domain.

This approach based on plans is similar to the position used in the expert model of ACT Lisp Tutor (Corbett, Anderson, & Patterson, 1990) for the programming tutor domain, the difference being that we apply it to a different domain. This difference in focus means that other aspects are
taken into account (e.g., varying the number of options available according to the student’s skill).
With Lisp Tutor the learners discover the productions of the language while they work on the
problem solving. In this system, the tutoring module acts as a problem solving guide but never
states the productions to be learned. In order to do this, it uses a series of correct production rules
for creating Lisp programs and the student model is built as a subset of these correct production
rules along with common incorrect productions rules (Holt, Dubs, Jones, & Greer, 1991).

We will now describe the characteristics considered in the student model, the monitoring of the
work carried out and how the system adapts to the user.

3.1. Student model

The user model consists of entities with information about the following categories:

– The student’s profile that defines the student’s role in the system. This profile stores restrictions
and obligations concerning the type of actions that the student should carry out. This is a con-
figuration aspect.

– The user’s interaction with the editor in order to plan the design. This information varies
dynamically according to the way in which the user interacts with the system to carry out a task.

– The sequence of actions that the students dynamically specify in their search for a solution to a
problem. Additionally, the time employed in the elaboration and the mistakes made in this pro-
cess are considered. Thus, each element in this sequence is defined by a unique identifier, the
moment the action is planned, the mistake associated to it (if there is one) and the action pre-
ceding the current action. This is,

\[
\text{item} ::= \text{id} <\text{id}><\text{time}><\text{action}><\text{prevAction}>
\]

where,

\[
\text{action} ::= \text{id} <\text{kind}><\text{area}><\text{section}><\text{element}>
\]

\[
\text{prevAction} ::= \text{id} <\text{action}>
\]

\[
\text{error} ::= \text{WRONG_ACTION}/\text{SEQUENCE_ERROR}/\text{DUPLICATE_ACTION}/\text{NON_OBLIGATORY_ACTION}
\]

3.2. Tutoring module

The information stored in the student model allows tracing by carrying out an automatic anal-
ysis of the way the user solves a design problem. This analysis can be carried out either dynam-
ically while the tool is being used or as a posterior process in order to draw general conclusions
(Redondo, Bravo, Bravo, & Ortega, 2003b).

In domotical design learning we propose that design problems with similarities in their solution
should be classified into categories. For each category (or generic kind of problem) a general strat-
egy of resolution is defined: the General Plan of Design (GPD). The GPD is a set of related actions
that can lead to the solving of a design problem. Some actions are obligatory while others are
optional. A precedence relationship imposing certain restrictions and obligations exists and there-
fore before carrying out an action others should have been previously achieved. Moreover, the
planning of an action may require others to be performed previously. For example, it is not possi-
ble to connect two elements if one of them has not yet been inserted in the design. However, if they
are linked, it will be necessary to define the characteristics of their relationship.

In a GPD there is a sequence of elements as follows:

\[
\langle \text{action} \rangle : \langle \text{type} \rangle : \langle \text{requirements} \rangle : \langle \text{influences} \rangle
\]

\langle \text{action} \rangle identifies an action, \langle \text{type} \rangle indicates whether it is obligatory or not,
\langle \text{requirements} \rangle indicates the actions that should be carried out before the current one and
\langle \text{influences} \rangle refers to those actions that should be carried out as a consequence of the cur-
rent action.

Nevertheless, each problem will be identified by a set of parameters. In some cases these are
fixed and in others variable. These parameters may condition the solution and allow the construc-
tion mechanism to be adapted to each student.

3.2.1. Guiding

The main objective of tracing the students’ resolution strategy is to determine if they are close
to a good solution. In many cases, there is not only one solution. We define the Optimal Plan of
Design (OPD) as a set of sequences of actions representing the possible solutions for a particular
problem. The system can modify the GPD according to the specification of the parameters that
characterize the problem. As a result, the OPD is obtained for the proposed problem. This consti-
tutes a set of resolution schemas for building the solution to the problem. There are different ways
of ordering the sequence of actions that can solve a domotical design problem. They should be
subject to the verification of some general properties (defined in the GPD). Other more specific
properties are reflected in the OPD.

By matching the information registered in the student model with the OPD, the system can
detect mistakes. Thus, if the student moves away from what the experts consider a good solution,
which is represented as a possible path of the OPD, the system can display warnings offering help
and reinforcement with the purpose of communicating to the students the mistakes they are making
and guiding them towards a better solution. This circumstance can make the student reflect in
search of an explanation and the correction of the mistake. This should lead to the promotion of
meta-learning situations.

Moreover, the system can modify the elements of the user interface including only those ele-
ments related to the best actions to be planned. Thus, the learner’s attention can be focused on
valid actions, and therefore, they are more likely to choose the best elements to build a valid and
good solution.

Fig. 1 shows a schema which summarizes the mechanism used for monitoring and guiding the
learner’s work. The beginning is the GPD for a generic kind of problem. The system can modify
the GPD according to the specification of the parameters that characterize the problem. As a
result, the OPD is obtained for the proposed problem. Additionally, the teacher sets up a help level
that affects the kind of messages that the system will provide in order to warn about or prevent the
mistakes that the solution proposed by the student can produce. Therefore, the selected and dis-
played help messages will be decided by an inference process originating from the OPD, the help
level and the design proposed by the student.
3.2.2. Adaptability

In order to increase the flexibility of the OPDs, a set of rules can be defined. These rules characterize the form in which a design strategy can be changed and the possible side effects of the changes.

We consider three sources of information that can cause a variation in the structure of the resolution of design problems: (a) the formulation and characteristics of the problem to be solved, i.e., its restrictions and requirements; (b) the design actions that the student is planning; and (c) the help level that the teacher determines for the resolution of this problem.

3.2.2.1. The characteristics of the problem. The parameters of a problem are classified into five groups: Identification Data, Environment Information, Housing Characteristics, Restrictions and Necessities of Design, Hypothesis and Simulation Cases. To each parameter in each group we associate a rule of the type

\[
\text{IF} \ <\text{condition}> \ \text{THEN} \ <\text{modifyAction}>
\]

This rule can modify the design actions and their structure. Thus, the rules associated to the definition of the problem will condition the characteristics of the OPD of each specific problem. In this way, the system generates specific design strategies for each case study. Thus, the adaptation depends on the domain.

3.2.2.2. Design actions. The design strategy that the students plan is defined according to a series of actions with a high level of abstraction (see Fig. 2). The order the student establishes when
approaching each generic action defines their resolution strategy. When reaching a state of the design at which different alternatives to continue are offered, the students choose one of them and discard the rest (see Fig. 3). They may choose a non-obligatory option, but selecting it will necessarily imply the performance of other actions.

When a state of the design offers several alternatives, each one is labelled with a numeric value. This value orders the alternatives from better to worse. By adding the values of the alternatives selected, a measure of the quality of the outlined solution is obtained. This allows us to guide the

Fig. 2. General actions of design in PlanEdit.

Fig. 3. Possible actions, actions chosen by the student and associated rules.
student to the best path, to determine the value associated to a good solution and to contrast it with the value associated to the solution that the student is planning. By so doing, the system can offer specific help according to the plan that the student is proposing.

3.2.2.3. Help level. When the teachers propose an activity in which a problem should be solved, they specify the help level. This level makes the use of scaffolding (Rosson & Carroll, 1996) possible, defining the degree of difficulty of the problem and the help level that the system offers to the student. This reinforces the process of problem solving. In this process new knowledge is built through a successive process of elaboration and integration.

The help level influences PlanEdit in two ways: the messages that will be displayed in order to guide the learners during their work (the messages displayed in relation to the help level matching the specifications are summarized in Table 1) and the set of possible actions that will be shown to the student (user interface). The learners have to choose the most appropriate action in each moment from the possible actions.

Fig. 4 shows the user interface of PlanEdit. This is structured in separate areas: the problem formulation, the list of tasks to carry out (tasks which give structure to the problem), the icon bars representing design actions/operators, the sequence of design actions already planned, the current action under construction and a set of buttons used to support several general functions.

The design actions that the student can choose are displayed in the user interface by means of icons in toolbars. They are grouped in four categories according to the components of an action: (a) the kind of action, (b) the management area, (c) the house plan and (d) the domotical operator.

The number and type of the elements present in each category will be different depending on the particular help level. That is to say, the quantity and kind of objects from which the student will have to choose will vary, as the tool adjusts the cognitive overload the user has to support. For example, if a problem that studies the thermal comfort area is being solved and the help level is high, the system will only show the actions and objects characteristic of this area. Otherwise, if the help level is low, it will show all the actions and the available objects, which will complicate the selection of the most appropriate ones. Thus, the interface adapts to the features of the activity. Depending on the characteristic aspects of the domain to study, the help to be displayed and the cognitive load in relation to the work material provided to the user will vary.

<table>
<thead>
<tr>
<th>Help level</th>
<th>Action checking</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Coherent precedence in the planned actions. (Obligatory/Optional) Membership to the plan of design. Actions list + current action =&gt; leading to a valid solution and refusing non correct actions</td>
</tr>
<tr>
<td>Intermediate</td>
<td>Coherent precedence in the planned actions. Advice about the actions list and the quality of the current action</td>
</tr>
<tr>
<td>Low</td>
<td>Coherent precedence in the planned actions</td>
</tr>
</tbody>
</table>
4. Design and implementation aspects

The analysis and design of PlanEdit has been carried out with the interactive participation of experts and teachers on the subject matter. This has implied an exhaustive evaluation of the tool in order to check whether it satisfies the needs involved in the learning situation. As a consequence of this process some user tasks have been identified and information in relation to the functionality of the system has been obtained. In the same way the initial mistakes in the conception of the system have been located and debugged from the first version (Redondo, Bravo, Ortega, & Verdejo, 2002).

The system is implemented in Java in order to facilitate interoperability as well as the adaptation to new platforms and environments offered (Cotter & Smyth, 2000). It is based on a client/server architecture optimized to provide good performance and efficiency.

The interaction of the users with the system and the work generated are stored in a database. This storage is achieved by using remote access to databases through the API JDBC 2.0 incorporated in the Java Virtual Machine. The database management systems used has progressively evolved since the first experiments. Having started with Microsoft Access in LAN (Local Area Network) environments, we are currently working with MySQL through Internet. Nevertheless, with this system it is also possible to work with the Oracle Database Management System when the administrator considers that the volume of data generated may be very high.
5. An experiment with students

In this section, we present the results of an experiment whose objective was to obtain some data that would allow us to confirm that our position could be effective for group learning of domotical design. This experiment was carried out with teachers and learners. The teachers randomly chose several students of Secondary Education without knowledge of domotical design to whom they taught Domotics. To do this, the teachers followed the protocol that we have outlined and used DomoSim-TPC and PlanEdit. We then tried to determine if the students had learnt specific aspects of the domain and techniques of group work. Thus, we have carried out an experiment in a real context in which twelve students were involved. These students represented 60% of the whole class in a profile of Professional Training in Secondary Education. They were all beginner students of Domotics.

The experiment was structured in several phases. In the first phase we asked the users to answer some questions in relation to design procedures (pre-test). The results of this test brought us to the conclusion that their level of knowledge about design, particularly where high complexity problems were concerned, was low. Next, the teachers proceeded to give some master classes where theoretical concepts in relation to the elements and design procedures used in Domotics were looked at.

The twelve students were organized in six groups of two (G1–G6). The teachers followed a procedure for the organization of the groups similar to the one used when the practical classes are given in a laboratory. That is, a list of groups is put up and the learners come to an agreement amongst themselves as to which group they wish to join. Also, the size of the groups (2 students per group) coincides with the size of the groups in the practical classes that take place in the laboratories. Following the advice of the teachers in the experiment, we thought that it would be better to maintain the same traditional structure and organization. Thus, the only change that we introduced in the experiment was the use of our system.

The learners were asked to carry out five design activities (A1–A5) of progressively increasing complexity. Thus, the first activity studied local aspects of some management area, while the last ones approached complex problems in the scope of a complete building. In this way not only the difficulty level but also the number of tasks to carry out increased. However, the help offered by the system in each activity progressively decreased, in contrast to the increase in the level of complexity.

We believe that a designer of domotical installations should be able to solve problems of service automation in buildings by means of installation design. The solutions to these problems are made up of design actions constituting a model and affecting domotical operators. The performance of correct design actions to solve a problem involves: (1) identifying the necessary operators; and (2) determining how the identified operators are integrated in the design of the installation. From this perspective, we suggest using the design action as a unit for evaluating the modeling work carried out. A designer can be considered skilled if for a design problem he/she is able to carry out many design actions whilst making few errors or needing little additional help.

With the registered data we focused on studying the evolution of the quantity of design work (or modeling work) developed in each activity and the number of errors made during this work. As errors we consider those mistakes observed in the final solutions and those situations in which the system intervened to avoid the performance of inappropriate design actions.
In order to graphically represent the previous information we define a parameter named *Quantity of Design Work* (QDW). This is expressed as the number of design proposals built with Plan-Edit in an activity multiplied by their average size (measured in number of design actions). In the same way, we have defined the *Number of Errors* (NE) made as the number of interventions and suggestions of the system during the elaboration process plus the number of mistakes observed in the final solutions the students have built.

Fig. 5 shows the evolution of QDW during the activities carried out. In this figure group G4 has not been included, since its participation level in the experiment did not pass the necessary threshold and did not contribute any significant information to this study. They simply decided not to work. In the other groups an important increase in the quantity of work developed is observed and the complexity of the activities also increases. We observe that group G3 in the activity A4 seems to have worked in a different way to the other groups. If we interpret this in a way which is coherent with the definition of QDW, we can say that group G3 has needed to carry out less design work (design actions) than the rest of the groups. Nevertheless, they have obtained a solution to the outlined problem. This circumstance is due to the fact that, according to other data that are not presented in this paper, the two members of the group adopted a method of working based on work distribution instead of on collaboration. That is to say, each student solved a part of the problem and there was little collaboration in the discussion stage. This meant that the initial proposals suffered few modifications. Nevertheless, we should point out that these initial proposals were valid and correct.

Fig. 6 shows the evolution of NE. As in the previous case group G4 has not been considered. We can observe that the number of errors that the students make tends to decrease significantly, in spite of the fact that the complexity of the work to carry out increases and the system offers less help. This represents the improvement in the design: the learners are learning and solve problems better each time.

In order to illustrate how the ability of the participants increased during the performance of this experiment we put the two previous parameters together. That is, we express the Ability of the participants in function of QDW and NE. Thus, the Ability is obtained as QDW/NE. We convert this parameter in a percentage in order to represent it in the graph shown in Fig. 7. In this figure, we can observe how the ability of all the participants increased progressively as they tackled new activities, despite these being more complex.
On the other hand, we can consider aspects of tool acceptance such as those approached by Bravo et al. (1999) with DomoSim (that is to say, the tool that PlanEdit improves). In that study, the acceptance of the installation of the new tool is evaluated according to the number of students that were able to complete the work (83%). In our experiment, 100% of the students finished the proposed activities. Nevertheless, it is necessary to highlight that the students of group G4 were not sufficiently motivated to begin the work proposed to them. Maybe if the organization of the groups had been more appropriate some other benefits of collaborative learning could have been exploited and this would have been avoided.

In the last phase, we tried to confirm what knowledge in relation to concepts of domotical design and what habits of group work the students had acquired. In order to do this, we looked at the application of methods based on questionnaires and interviews. However, we believed that it would be more realistic to observe how they performed in a similar situation to that in which they would be when required to put into practice the acquired knowledge. Thus, we carried out a test (post-test) to determine if the students would be able to manage in real situations and with authentic materials. We visited a laboratory where advanced installations of automation of services for buildings were being developed. The students immediately identified the elements that were being
In order to provide adaptability features for a computer-supported collaborative learning tool, we have proposed a method based on the use of generic and specific plans. This proposal is applicable in situations of Problem-Based Learning and Collaborative Problem Solving as instructional paradigms for design learning.

We have developed an adaptive editor of design strategies with the aim of facilitating the learning of building models. In this paper, we have focused on the individual perspective of learning, although this tool is integrated in an environment supporting collaborative learning called DomoSim-TPC. This tool supports the performance of learning activities enhancing discovery and experimentation. The learning tasks consist of the resolution of design problems. The student should build a model able to satisfy the specifications of a design problem. We have represented a general procedure to solve problems as a General Plan of Design (GPD). For each problem the system automatically infers the Optimal Plan of Design (OPD) from the GPD. We have considered several aspects of the users, such as their procedural strategy (student plan) to solve a problem as a history trace of the path followed and their behaviour interacting with the system. With this information, the tool varies the cognitive load of the user interface and adjusts the alternatives present at each stage in the resolution process. This adjustment shapes the presence of reinforcing help and the concealing/prohibition of alternatives to guide the learner through the resolution process (adaptation of the path).

In our proposal, the path the learner outlines has to match one of the plans. However, the outstanding characteristic of our tool in contrast to hypermedia systems is that the learner does not follow a content-based path. On the contrary they build a path as a plan or solution strategy to a proposed problem. The system adapts to the characteristics of the problem and the solution that the students propose, trying to guide them so that they can achieve their objective. Techniques similar to those which we present here have already been applied to other domains like programming tutors. As an innovation we outline their application in a radically different domain. As a consequence we have incorporated some new strategies such as the variation of the number of alternatives or different options in function of the student’s level.

The experiment presented here confirms that our proposal can be effective as a tool to increase the design ability and the involvement of students, and therefore, for the learning of domotical design. Additionally, it promotes the acquisition of group work habits.

Our proposal is applicable to any domain where the resolution of design problems in a structured way is approached. This resolution can be specified in terms of the design actions to build a model. These actions can be inferred from the user interaction with the tools traditionally used in that domain.
Acknowledgements

This work was partially supported by the Junta de Comunidades de Castilla – La Mancha and the Ministerio de Educación y Ciencia in the projects PBI-02-026 and TIC2002-01387.

References


